Orchestration and Choreography: Standards, Tools and Technologies for Distributed Workflows

Stephen Ross-Talbot

Pi4 Technology, London, UK and W3C, Geneva, Switzerland

Abstract

Workflow and the terms orchestration and choreography are confusing terms at the best of times. They all play a role in the structuring of systems and they are all a result of the need for abstraction and structural clarity of systems. There has been a lot effort over the last three years on both orchestration and choreography and tools and standards that govern their interoperability are close to maturation.

In this lecture we shall look at workflow, orchestration and choreography with respect to the wider notion of Service Oriented Architecture and explain how they can be used, what they do and how they relate to one another. And all the time we shall keep in mind the notions of abstraction, structural clarity and domain relevance.

During the course of this lecture we shall look at how these notions complement each other and how they can best be used to describe and execute workflows in a distributed service oriented architecture setting.

We shall use some tools to demonstrate how they work and present some examples that have relevance to the domain of workflow in bioinformatic systems.

Introduction

The promise of reuse in software systems has been a common theme in commercial computer science since the advent of [SmallTalk] and [Ada]. Reuse became a popular topic in the mid 1990's with the advent of [C++]. Indeed reuse was a big claim of object-orientation along with it's companion productivity. At the same time object-orientation came to the fore so did service based computing. We first saw it with the Esprit programme through projects such as the [SoftwareBus]" and then with [CORBA]. Today we see it with Web Services and the generic concept of [Service Oriented Architecture] (SOA).

Workflow techniques have now come to the fore within this push for reuse and productivity. The problem we face is how to reuse services, which do things, and then link their results to other services and so deliver some result. This can be some commercial result like booking a holiday or indeed a scientific one in which scientific services are used as part of an in-silico workflow for experimentation support.

What held workflow back for a long time was the cost of integration of services into the workflow. Describing the workflow has never been easier but giving it meaning by linking services together and their results is less easy. This is the promise of Service Oriented Architecture. It provides an infrastructural plane in which we can describe linkage and thereby gain reuse of common services in different contexts.

In this lecture we shall look at workflow, orchestration and choreography in a Service Oriented Architectural plane. Having defined the terms we shall look more deeply at what they do and how they do it. In particular we shall focus on bringing out the different levels of abstraction that they support and provide structural clarity over them.

Finally we shall look at some tools, restricted to choreography, and demonstrate the abstractions and structural clarity that underpins choreography and do so within the domain of relevance.

The aim of this lecture is to provide clarity and guidance as to how best one might describe and execute workflows in a distributed service oriented architecture setting.

.IDefinitions

Workflow is an IT technology that uses electronic systems to manage and monitor business processes. It allows the flow of work between individuals and/or departments to be defined and tracked. Workflow software helps you automate a range of business tasks, and electronically route the right information to the right people at the right time. Users are notified of pending work, and managers can observe status and route approvals through the system quickly.

Service Oriented Architecture (SOA) is an architectural approach for the implementation and delivery of services, where service is some well formed computation process. There is nothing new about SOA. Rather it is a set of guiding principles which are supported by a range of standards that make it possible to define, implement and deliver a service in a uniformed way so that it can be reused in different contexts. The dominant set of standards are those that are known as WS-*. Included in these are [WSDL], [SOAP], [http], [UDDI], [SAML], [WS-BPEL], [WS-CDL]. There are more but for the purpose of this lecture these are the important ones.

SOA is not just about Web Services and so the WS-* stack of standards and the infrastructure that they suggest is not the only way to deliver a SOA. SOA can be based on [Java] using Java interfaces instead of WSDL and using RMI instead of SOAP and HTTP.

Orchestration is all about recursive composition of services. This typically includes both externally and internally visible message exchanges, any additional business logic needed to record externally and internally visible state and is done from the perspective of the controlling service. WS-BPEL is a standard for orchestration and is used to compose new web services from existing web service; WSDL and the rest of the WS-* stack are pre-requisites.

What orchestration, and therefore WS-BPEL, does is to enable a user to define a new service from existing services and present the result as a service. This is why it is often described as a language for recursive composition. The new service is a first class service in an SOA world and has an execution.

A **choreography** is a description of the peer to peer externally observable interactions that exist between services. The interactions are described from a global or neutral point of view and not from any one services perspective. WS-CDL, as an standards exemplar, is based on WS-* but is not required to be so. Thus WS-CDL has a role as a description language for a common behavioral contract in a general SOA idiom. Whereas orchestration dictates an execution style (that of the brokering service) choreography languages do not. Rather they describe the common observable interactions (the messages flows) between services without mandating any execution style. And whereas a language like WS-BPEL, for orchestration, is itself executable, choreography can be used to generate the necessary behavioral contract for each of the peers, further non-observable logic (code) maybe required to manifest the full set of service implementations.

SOA, Workflow, orchestration and choreography.

In a SOA services can be software agents as well as human agent. It is when we include a human agent as a service that we start to see that orchestration and choreography are different ways of defining workflows. In the case of orchestration we use a service to broker the interactions between the services including the human agents and in the case of choreography we define the expected observable interactions between the services as peers as opposed to mandating any form of brokering.

What SOA provides is a common infrastructural idiom that decouples services so that they can be reuse. This is what we call loose coupling. There are 3 main principles to any SOA which are important because they provide the necessary architecture over which a workflow can operate.

1. Statelessness requires that services do not hold state across invocations – this way we can leave binding from one service to another within a workflow to the infrastructure.
2. Interfaces are required to be externally visible so that we can understand what linkages are available – a workflow structures these into valid interactions between services to achieve some goal.
3. Discoverability requires that services can be located based on some discriminating factor – which can be used by a workflow to locate services based on security, transactionability, and even behavior.

Workflow can be described using WS-BPEL and other orchestration languages and can also be described by a choreography language such as WS-CDL. Which one you choose really depends on what you want to achieve. It is often the case that both are needed and both provide value. For example one might describe the workflow as a choreography and realize the workflow as a set of executable workflows that are linked by design. In this scenario the services may be orchestrated (using WS-BPEL compliant tools) such that the external observable behavior is defined (and generated) from an agreed choreography (using WS-CDL). Thus the choreography is a blueprint for the overall system and the orchestration is a means of realizing the system without introducing an overall broker-service. One may think of this as islands of orchestration in which the islands interact based on the blueprint.

Relevance to bioinformatics

The relevance of workflow, orchestration and choreography within a SOA is based on three things. The underpinning formalism turns out to be relevant to both the IT community and bioinformatics. The use of SOA turns out to be a major factor in gaining the necessary reuse to conduct in-silico experiments. The cost and regulatory pressures become more manageable and transparent as a result.

.IFormalisms

The use of the [pi-calculus] as an underpinning formalism for workflow be it WS-BPEL or WS-CDL is interesting in and of it's own right. The scale invariance of the underpinning formalism sets it apart.

The very same formalism that is showing efficacy in the design of service-oriented architectures is also showing predictive power in the analysis of chemical, biochemical and biological systems by applying [stochastic pi-calculus] to modeling a number of different biological systems from metabolic pathways to signal transduction pathways to intercellular processes like inflammation.

Both applications face an intriguing set of challenges about how to model wide-scale distribution. Both applications face an interesting set of challenges in how to relate protocols at one level to protocols at another. For example, in the physical world we have

Agents`	Communication Events
Small molecules	Electron Sharing
Proteins	Protein-protein interaction, binding, phosphorylation, methylation, etc
Cells	Material consumption, environmental sensing, etc

while in the world wide web we have

Agents	Communication Events
Network services	Tcp/ip read/write
Email client/server	Smtp read/write
Browser/server	http read/write
Financial apps	Reliable messaging

Relating these protocols at very different levels of the "stack", assumptions about location, failure, etc. are major challenges facing the process algebras of which the pi-calculus is the root.

.IISOA in bioinformatics

SOA in the lab and between the labs is highly relevant to the use of successful workflows described using WS-CDL. Biological experiments are so complex in the post-human-genome-era that they can be said to be workflows. The problem is that the devices involved in these experiments speak in a variety of non-self-describing formats and are not currently thought of as agents/services and so do not have documented protocols or programmatic interfaces

The emergence of the pathway databases (e.g., KEGG, which publishes a WSDL and ought to publish a set of WS-CDL description, Biocyc, Reactome, etc) clearly indicates that biological datasources are best seen as webservices.

.IIIVolatility in process

The human organizations in biological research institutions, e.g. pharmaceuticals, are undergoing a new kind of volatility -- needing to be more flexible about outsourcing various aspects of a process. Outsourcing of this kind demands clear expectations of the protocol between the client and service organization. Human organizations need to be able to be seen as services in a larger service-oriented-architecture.

Nowhere is this more clear than in clinical trials. The clinical trial protocol (what gets measured in whom at what time) is a tiny little sliver of the actual process involved in carrying out the trial. Each clinic has policies and procedures that must be respected to implement the protocol. Codifying these processes will help immensely in addressing the failure rate of these trials. More importantly, in the wake of disasters like Vioxx, the desire for transparency and accountability on the part of the public and regulatory bodies makes codification of the processes a necessary step. In this regard WS-CDL is an ideal solution because it is peer to peer and has no central point of execution.

As a consequence of the human-genome project a more predictive, preventive medicine will emerge in which individual genomic data and history is included in the clinical protocol. Therefore, the complexity of these protocols will go through the roof. If there is not a compositional way to address the design, analysis and execution of these protocols, it will become unmanageably complex. This is why the pi-calculus has a direct benefit, because of its scale invariance it can model in the small and the large.

WS-CDL – describing peer to peer relationships

The Web Services Choreography Description Language (WS-CDL) is an XML-based language that can be used to describe the common and collaborative observable behavior of multiple services that need to interact in order to achieve some goal. WS-CDL describes this behavior from a global or neutral perspective rather than from the perspective of any one party.

Services are any form of computational process with which one may interact, examples are a buying process and a selling process that are implemented as computation services in a Service Oriented Architecture (SOA) or indeed as Web Services. The distinction between SOA and Web Services is that the former may not have a Web Services Description Language interface (WSDL) whereas the latter will do so. This WS-CDL can play the same role for both SOA services and Web Service Services.

Common collaborative observable behavior is a phrase we use to indicate that the behavior of a system of services can be externally observed by a third observing party, for example the interaction between buyer and seller services. Each service has an observable behavior that can be described today using WSDL or some other interface description language (e.g. Java). Such observable behavior is described as a set of functions that the service offers coupled with error messages or codes that indicate failure of some sort, this can be system failure (e.g. a server reboots) or a failure at some business level (e.g. the stock availability is less that is being ordered). If we used abstract BPEL along with WSDL we can also describe the valid sequences of functions that cannot be done with WSDL or Java alone. We refer to this as "observable behavior" because for any service we can only describe and then use that behavior that is visible or observable (e.g. we can observe a credit rejection but we may not observe the reasons why that rejection occurred, the former is observable and the latter is non-observable). The common collaborative observable behavior is that subset of the behavior of the set of services required to achieve a goal (e.g. buying some Chocolate Oreos) that necessitates integrating the behaviors of the individual services and choreographing them (e.g. the sequences between them, what can be parallelized, what dependencies they may have on each other). What WS-CDL allows one to do is describe the effective ordering constraints across the services and so describe the common rules and common behavior in order for the services to collaborate. This is why we call it "common collaborative behavior".

The global or neutral perspective, also referred to as "a global model", ensures that the common collaborative observable behavior is not biased towards the view of any one of the services. Instead it describes the entire collaborative observable behavior of all of the services as peers such that no one service can be said to exert any control over any other service. In effect it described the services as a complete distributed application in which each service plays a distinct role and has distinct relationships with its peer services.

In WS-CDL the mechanisms for describing the common observable behavior range from specific information alignment (e.g. when a buyer and seller record the fact that an order has been accepted in variables that reside at the buyer and at the seller), interaction (e.g. when a buyer requests a price from a seller and receives a price as a response from the seller) and a declaration of interest in the progress of a choreography (e.g. has a the bartering choreography between buyer and seller "started" or has it "finished"). In the first two cases synchronization is explicit and visible as a business related activity (e.g. the observable recording of information and it's alignment and the description of an information exchange between a buyer and seller) and in the last case (e.g. choreography has "started" or "finished") it is implicit based on the progress of a choreography and not any business relationships.

.IThe benefits of using WS-CDL

WS-CDL can be used to ensure interoperability within and across domains of control and so create solutions across domains of control.

WS-CDL can be used to ensure that the total cost of software systems in a distributed environment, within a domain of control and across the world-wide-web is lowered by guaranteeing that the services that participate in a choreography are well behaved on a continuing basis.

WS-CDL, as with many workflow-based solutions, provides greater flexibility because the observable interactions are out-boarded. This means that the choreography can be changed independently of the services and yet can continue to guarantee interoperability.

WS-CDL, being based on the pi-calculus, can be used to show that services will behave correctly based on advanced static behavioral type checking.

WS-CDL can be tested for equivalence based on what we call bi-simulation. This is of fundamental import to regulators who can use it to prove conformance of an implementation of a workflow to a description in WS-CDL of that workflow.

.IIThe Structure of WS-CDL

WS-CDL is a layered language. It provides different levels of expressability to describe a choreography. These levels are shown below in Figure 1.

At the top most level for any WS-CDL there is a package that contains all other things. All choreographies described in WS-CDL will include as a minimum a set of Roles that are defined as some sort of behavior (i.e. a WSDL description) and so represent our WS-CDL notion of a service, Relationships between those roles, Channels used by roles to interact and a Choreography block that uses channels to describe Interaction. What the choreography describes at this level is a basic set of typed and unambiguous service connections that enable the various roles to collaborate in order to achieve some common goal.

Adding further ordering rules through Structured composition allows Interactions and Choreographies (which are just logical groupings of interactions) to be combined into sequences, parallel activities and so on.

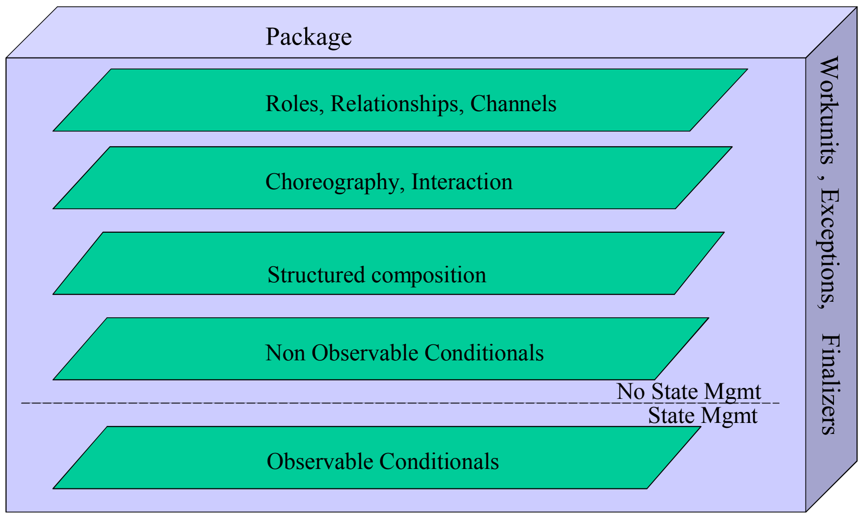
Package
Roles, Relationships, Channels
Choreography, Interaction
Structured composition
Non Observable Conditionals
No State Mgmt
State Mgmt
Observable Conditionals
Workunits , Exceptions, Finalizers

Figure 1: Layered structure of WS-CDL

Adding Non-Observable Conditionals makes it is possible to model branching based on observing changes in the interactions that occur (e.g. one might observe an exchange between a buyer and a seller which is said to be terminated when a “completed” interaction is observed – an existential predicate returning true when a “completed” interaction occurs.

If we have no observable conditionals then it is not necessary to perform any explicit state management at the roles that are interacting because we have not needed to express any explicit computation (e.g. totalOrderValue EQUALS expectedOrderValue) required of an observable condition. By this we mean that none of roles used in choreographies at this level have any notion of shared state, rather they observe interactions that are visible and use the observations to determine their state with respect to the other roles.

Some business protocols are defined with specific business rules visible. These constitute shared knowledge between the roles concerned, for example we might terminate an order completion between a buyer and a seller when we calculate that the items delivered match the original order. The business rule in this example is the shared constraint that buyer_quantity equals completion_quantity. At some level the roles must have some shared knowledge of both variables and their values. When business rules of this nature become part of the business protocol such Observable Conditionals can be added into a choreography and this now implies state management is needed.

State management requires an amount of machinery to ensure that state is known globally when needed or at least between roles when needed. The machinery required to deliver this in practice is some coordination mechanism that will ensure data is delivered to all roles that require it.

An Example

Having provided a framework for peer-to-peer descriptions of workflow and having outlined the benefits of so doing the rest of this lecture will present some tools and an example with which to further illustrate our topic.

Acknowledgements

Our thanks to Greg Meredith of Djinnisys who provided much contextual guidance for this paper.

References

[SmallTalk] "**What is Smalltalk?**"Peter William Lunt http://www.smalltalk.org/smalltalk/whatissmalltalk.html

[Ada] "The History of the Ada Programming Language", Ryan Stansifer http://www.cs.fit.edu/~ryan/ada/ada-hist.html

[C++] "The C++ Programming Language" Bjarne Stroustrup http://www.research.att.com/~bs/3rd.html

[SoftwareBus] Schafer, W. and Weber, H., The ESF-Profile, in Handbook of CASE,P.Ngand R. Yeh, Editors, Van Nostrand Reinhold, New York, 1989.

[SoftwareBus] Christer Fernström, The Eureka Software Factory: Concepts and Accomplishments, Proceedings of the 3rd European Software Engineering Conference, p.23-36, October 21-24, 1991

[CORBA] "Corba Basics" Object Management Group

http://www.omg.org/gettingstarted/corbafaq.htm

[Web Services] "Web Services Architecture", W3C

http://www.w3.org/TR/ws-arch/

[SOA] "What is Service-Oriented Architecture?" Hao He

http://webservices.xml.com/lpt/a/ws/2003/09/30/soa.html

[WS-CDL] "Web Services Choreography Description Language Version 1.0" Kavatzas et al.

http://www.w3.org/2002/ws/chor/edcopies/cdl/cdl.html

[WS-BPEL] "OASIS Web Services Business Process Execution Language TC "

http://www.oasis-open.org/committees/wsbpel/charter.php

[WSDL], "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer"

http://www.w3.org/TR/2005/WD-wsdl20-primer-20050803/

[SOAP], "SOAP Version 1.2 Part 0: Primer"

http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[http], "HTTP - Hypertext Transfer Protocol"

http://www.w3.org/Protocols/

[UDDI], "OASIS UDDI Specification TC"

http://www.oasis-open.org/committees/uddi-spec/charter.php

[SAML], "OASIS Security Services TC"

http://www.oasis-open.org/committees/security/charter.php

[Java] "The Java Language Specification"

http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf

[Workflow in bioinformatics] "Comparing Workflow in eScience and eBusiness" Mark Greenwood, 26th September 2003

http://twiki.mygrid.org.uk/twiki/bin/view/Mygrid/WorkFlowDifferences

[Pi-Calculus] "The Polyadic pi-Calculus: A Tutorial" Robin Milner, Springer-Verlag, 1993

[Stochastic Pi-Calculus] "Application of a stochastic name-passing calculus to representation and simulation of molecular processes" Corrado Priami , Aviv Regev, Ehud Shapiro, William Silverman

Information Processing Letters Volume 80, Issue 1 (October 2001)

[WS-CDL Tools] www.pi4tech.com and www.pi4soa.org

[Pi-Calculus tools for bioinformatics] http://www.djinnisys.com/