Web Service Choreography Description Language (WS-CDL): Goals and Benefits

Meenu Rani1, Amit Kumar Chawla2, Shalini Batra3
1ME (Computer Science), meenu_gyl@yahoo.co.in
2ME (Computer Science), amitkchawla@yahoo.com
3Lecturer, sbatra@tiet.ac.in

Computer Science and Engineering Department, Thapar University, Patiala (147004), Punjab, India

Abstract

The Web Services Choreography Description Language (WS-CDL) is an XML [1]-based language that describes peer-to-peer collaborations of participants by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal. The Web Services [2] specifications offer a communication bridge between the heterogeneous computational environments used to develop and host applications. The future of E-Business applications requires the ability to perform long-lived, peer-to-peer collaborations between the participating services, within or across the trusted domains of an organization. The Web Services Choreography specification is targeted for composing interoperable, peer-to-peer collaborations between any type of participant regardless of the supporting platform or programming model used by the implementation of the hosting environment. In this paper we will be discuss what is the need of a choreography language and the benefits and goals of the WS-CDL.

1. INTRODUCTION

Choreography defines the rules and interactions of collaboration between two or more business entities. Choreography is described from the perspectives of all parties, and defines the complementary observable behavior between participants of collaboration. A common view, in essence, defines the shared state of the interactions between business entities. This common view can be used to determine specific deployment implementations for each individual entity.

In real-world scenarios, corporate entities are often unwilling to delegate control of their business processes to their integration partners. Choreography offers a means by which the rules of participation for collaboration can be clearly defined and agreed to, jointly. Each entity may then implement its portion of the choreography as determined by their common view.

A Choreography allows constructing compositions of Web Service participants by explicitly asserting their common observable behaviors, where synchronized information exchanges through their shared contact-points occur, when the commonly defined ordering rules are satisfied.

2. OVERVIEW

The aim of Web Service Choreography is to provide means for describing behavioral interface of a single Web Service, and for specification of interaction protocols for Web Service collaborations. The notion of Choreography is intended to describe how to consume Web Services, i.e. making use of their functionality.

WS-CDL is a language for specifying peer-to-peer protocols where each party wishes to remain autonomous and in which no party is master over any other – i.e. no centralization point. The description of a peer-to-peer protocol is grounded in what we term an ordered set of interactions, where an interaction is defined loosely as an exchange of messages between parties.

It is essential in understanding Web Services Choreography Description Language (WS-CDL) to realize that there is no single point of control. There are no global variables, conditions or work units. To have them would require centralized storage and orchestration. WS-CDL does permit a shorthand notation to enable variables and conditions to exist in multiple places, but this is syntactic sugar to avoid repetitive definitions. There is also an ability for variables residing in one service to be aligned (synchronized) with the variables residing in another service, giving the illusion of global or shared state.

It is also important to understand that WS-CDL does not distinguish between observable messages from applications, that might be considered as application or business messages, from the infrastructure upon which an application is based, that might be considered as some form of signal. In WS-CDL all messages are described as information types and have no special significance over each other. All that WS-CDL describes is the ordering rules for the messages, which dictate the order in which they should be observed. When these ordering rules are broken WS-CDL considers them to be out-of-sequence messages and this can be viewed as an error in conformance of the services that gave rise to them against the WS-CDL description.

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 and we call a complete WS-CDL description a global model.

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 computational services in a Service Oriented Architecture (SOA) [3] or indeed as a Web Services implementation of an SOA. Because WS-CDL is not explicitly bound to WSDL [4] it can play the same global model role for both SOA services and Web Services, that is it is possible to use WS-CDL to describe a global model for services with no WSDL descriptions (perhaps they just have Java interfaces) as easily as it is to describe services that do have or will have WSDL descriptions. The way in which WS-CDL can be used without WSDL descriptions is however implementation dependent.

Common collaborative observable behavior is the phrase we use to describe the behavior of a system of services, for example buyer and seller services, from a global perspective. 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, possibly with parameters, that a service offers coupled with error messages or codes that indicate failure along with the return types for the functions offered. If we used abstract BPEL [5] along with WSDL we can also describe the valid sequences of functions from a single services perspective (i.e. the service we are describing), which is not possible with WSDL or Java alone. We refer to this set as the “observable behavior” for a service. This level, the service level, of “observable behavior” does not describe behavior of a system of services because it only deals with a single service. The composition of a set of “observable behaviors” at a service level is what we call the common collaborative observable behavior. The composition is not simply the set of observable behaviors at the service level operating together because such a composition requires further description of the dependencies that the set of services exhibit in order to interoperate correctly. If we captured the ordering rules for a set of service then we would have the common collaborative observable behavior fully specified. This is what WS-CDL is for.

Individual service behaviors can be used in the composition of wider collaboration in which a set of services with their own behaviors could be effectively used. In order to do so a global model that described the peer to peer observable interactions of such a set of services is required to ensure that the services will in-fact cooperate to a commonly understood script. That script is the global model and that script is what WS-CDL is used to describe.

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 as peers the entire collaborative observable behavior of all of the services 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.

One may think of WS-CDL as a language for describing the observable activities of a set of services some of which are synchronized through some common understanding realized by a specific business interaction between the services or by a declaration of interest in the progress of one service by another (e.g. has the buyer accepted the price offered by the seller). The least interesting scenario is one in which WS-CDL can be used to describe a set of services that never synchronize at all; that is there is no observable relationships and no statement of an unobservable relationship that exists between the services. In this case the services perform choreography, but effectively on different stages and thus need no form of coordination. In all other cases the synchronization is what makes life interesting (e.g. a buyer seller choreography coupled with a seller credit check choreography or indeed a seller shipper choreography).

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 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.

3. Goals

The primary goal of the WS-CDL specification is to specify a declarative, XML based language that defines from a global viewpoint the common and complementary observable behavior specifically, the information exchanges that occur and the jointly agreed ordering rules that need to be satisfied.

More specifically, the goals of the WS-CDL specification are to permit:

- **Reusability.** The same choreography definition is usable by different participants operating in different contexts (industry, locale, etc.) with different software (e.g. application software)
- **Cooperation.** Choreographies define the sequence of exchanging messages between two (or more) independent participants or processes by describing how they should cooperate
- **Multi-Party Collaboration.** Choreographies can be defined involving any number of participants or processes
• **Semantics.** Choreographies can include human-readable documentation and semantics for all the components in the choreography

• **Composability.** Existing choreographies can be combined to form new choreographies that may be reused in different contexts

• **Modularity.** Choreographies can be defined using an "inclusion" facility that allows choreography to be created from parts contained in several different choreographies

• **Information Driven Collaboration.** Choreographies describe how participants make progress within collaboration, through the recording of exchanged information and changes to observable information that cause ordering constraints to be fulfilled and progress to be made

• **Information Alignment.** Choreographies allow the participants that take part in choreographies to communicate and synchronize their observable information

• **Exception Handling.** Choreographies can define how exceptional or unusual conditions that occur while the choreography is performed are handled

• **Transactionality.** The processes or participants that take part in choreography can work in a "transactional" way with the ability to coordinate the outcome of the long-lived collaborations, which include multiple participants, each with their own, non-observable business rules and goals.

4. **BENEFITS**

The Web Services Choreography working group believes that all uses of a choreography description necessitate the existence of a standardized language for the description of choreographies. The benefits of such an approach:

• Promote a common understanding between WS participants.

• Automatically guarantee conformance.

• Ensure interoperability.

• Increase robustness.

• Generate code skeletons.

• More robust Web Services to be constructed.

• Enable more effective interoperability of Web Services through behavioral multi-party contracts, which are choreography descriptions.

• Reduce the cost of implementing Web Services by ensuring conformance to expected behavior.

• Increase the utility of Web Services, as they will be able to be shown to meet contractual behavior.

5. **CONCLUSION**

The Web Services standards ‘stack’ has already become disconcertingly complex, although small subsets of it (such as WSDL alone) can be used successfully for many applications. The full panoply of specifications is required only for inter-enterprise e-business transactions. Until now, there hasn't been a standard way to describe the flow of messages exchanged by a Web service participating in a choreographed interaction with other services. WS-CDL can be used to ensure interoperability within and across domains of control to lower interoperability issues, and create solutions within and 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 choreography are well behaved on a continuous basis. Both of these benefits translate into greater up time and so increase top line profits. At the same time they translate into less testing time and so reduce cost of delivery, which decreases bottom line costs.

**REFERENCES**


