Web Service – Composite Application Framework

By Verdaguer Olivella, Joan

Table of Contents

1 Abstract 3

2 Background 4

3 The purpose of WS-CAF 4

4 The WS-CAF 4

4.1 WS-CTX 5

4.1.1 ALS 7

4.2 WS-CF 7

4.2.1 Comparison with WS-C 8

4.3 WS-TXM 9

4.3.1 ACID Transactions 9

4.3.2 LRA Transactions 11

4.3.3 BP Transactions 12

5 Differences between WS-CAF, WS-T and BTP 14

6 Conclusions 14

7 References 16

1 Abstract

In the lasts years, with a high development in the communication technologies, lots of business have started to change the way they do the transactions between them. Nowadays it is very common for businesses to implement some transactions with computer systems. Thus, gaining speed in the transaction and reducing the amount of work that humans should do (between other advantages). More deeply, one of the common ways to implement these transactions is with web services, a technology that has lots of advantages.

But there have appeared lots of web services that work in different ways, use different output and are suitable for different kind of transactions. Because of this, a business can face difficulties when it wants to develop a big system to make a complex transaction which involves interaction with other web services. Although having very differences between those web services, the system should be able to coordinate them properly. And this includes giving/retrieving information about the current state of the transaction, receiving different kind output from each service, and using the different kind of features they have (which may be particular in each web service), etc.

This paper is about an initiative called Web Services Composite Application Framework, a set of standard specifications purposed by the organization OASIS. It defines a Framework that tries to make it much easier to compose different web services into a bigger application, reducing the cost of its implementation and increasing the interoperability. We will explain the three different standards this framework is composed of, which are the WS-Context, the WS-Coordination Framework and the WS-Transaction Management, as well as which are the benefits of each of them. We will also mention the differences of this standard with the Web Service Transaction and Business Transaction Protocol standards, which are also designed to handle transactions between businesses.

2 Background

Nowadays, to develop an effective application that can perform a business transaction we might probably need to interact with other services or applications from other business. An application of that kind is called a composite application, since its processes are a composition of other processes.

One of the difficulties faced when someone is trying to build a composite application is the wide variety of protocols when using web services. Although the variety of protocols may make one application fit more the transaction it should represent, it comes with the repercussion that such application will be much more difficult to implement.

3 The purpose of WS-CAF

The purpose of the Web-Service Composite Application Framework (WS-CAF) is to provide a Framework capable of making the implementation of composite applications easier. For this, it aims for making the application developer not care about the details of the protocol and coordination details and its implementation.

Although there have already been several attempts to make this possible, none of the ones before had been designed in such a way to be useful for an enough variety of transactions.

4 The WS-CAF

The WS-CAF is composed of three standard specifications. Those are the following:

- Web Service Context (WS-CTX).
- Web Service Coordination Framework (WS-CF).
- Web Services Transaction Management (WS-TXM).

Those specifications are meant to be used together in stack (WS-TXM over WS-CF over WS-CTX).

WS – CAF works on SOAP messages Web Services, so it also uses WSDL messages to locate the corresponding Web Services.

The WS-CAF is an standard specified by the OASIS standards organization.

4.1 WS-CTX

Web Service-Context is a standard meant to share data about the state of the transaction between the different web services an application is composed of. The data sent includes the context identifier, the type of the context, and a timeout value, and a list where the application can add attributes at will. This data can be modified by the web services implied in the composite application as the transaction goes on.

The context information has an XML structure, and is sent in the header of the SOAP messages exchanged in a web service invocation.

Here follows an example (extracted from Mark Little, Eric Newcomer and Greg Pavlik., "WS-CAF: Contexts, Coordination and Transactions for Web Services"):

```
<ContextType> MyContext </ContextType>
<context-identifier>
www.webservicestransactions.org/example/Middlewar
e2004ContextExample
</context-identifier>
. . .
. . . mustUnderstand=true
. . . mustPropagate=true
. . .
<child-contexts>
        <child-context>
                <user-name> EricNewcomer </user-name>
                <password> ******** </password>
        </child-context>
        <child-context>
                <database-name> SQL-DB </database-name>
                <file-name> Index-S-file </file-name>
                <display-address> PocketPc25 </displayaddress>
        </child-context>
```

```
        <child-context>
                <transaction-type> BusinessProcess </transaction-type>
                <transaction-mode> Required </transactionmode>
        </child-context>
</child-contexts>
```

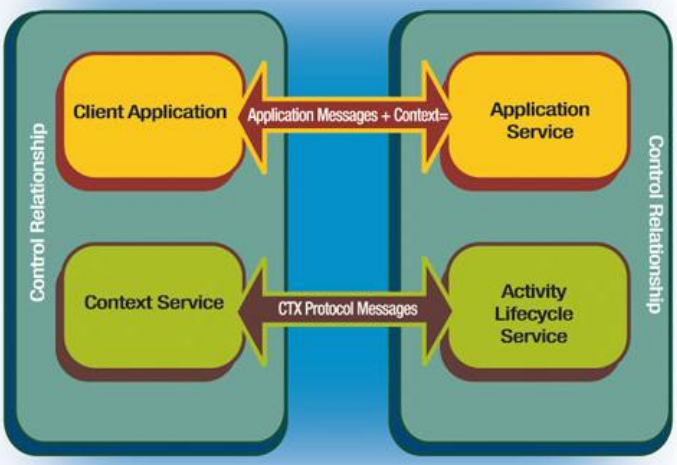
As we can see, there are three main elements:

- <ContextType>
- <context-identifier>
- And <child-contexts>

The first two XML attributes regard information about the structure of the context itself. The last one contains a list of "<child-context>" where into each one of these there is a value (or values) for some attribute that is needed to be sent as context information. XML syntax makes it easier to add these kind of attributes because of its hierarchy.

There are also two attributes in this example (*mustUnderstand* and *mustPropagate*). The first one means that the comprehension of this context is necessary to proceed with the request, and the second one means that the information in this context might be necessary to other services which are also in the composite application.

The following picture represents the message exchange between the client application and the service it uses as well as the message exchange regarding the context management (extracted from [4]):



As we can see in this picture, the client application exchanges the context with the service it uses within the header of the messages it sends to ensure its propagation. There is also another service called Activity Lifecycle Service (ALS) interacting with the context service.

4.1.1 ALS

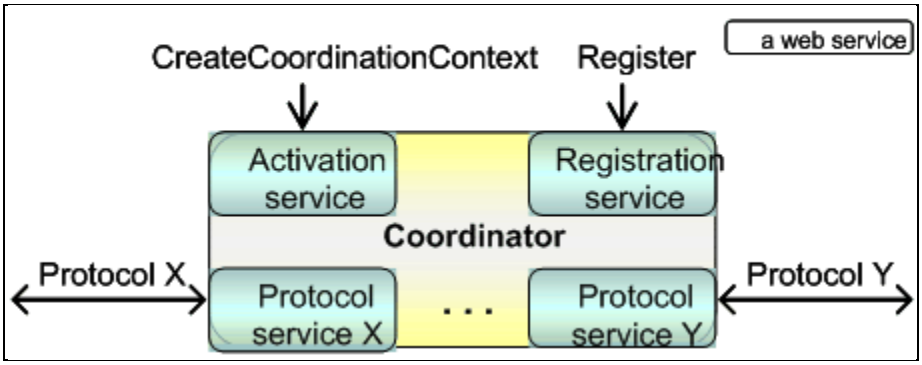
While a context service is held by the client application, an ALS is maintained by the services this application uses. Basically, each service can register an ALS to the context service of the application. Then, the context service of the application calls every registered ALS to obtain context information from these services, and then uses it to create the actual context information that is propagated through the SOAP header.

4.2 WS-CF

The Web Service – Coordination Framework acts as a coordinator between the different participants in one transaction. This means, it is in charge of achieving consensus between itself and the different participants regarding the coordination protocols they will use, and it is in charge of propagate the transaction context as well.

Different Web Services (and other composite applications as well) can register themselves as participants in a WS-CF.

The following picture represents the behavior of a Coordinator (extracted from [11]):



A WS-CF instance can also be registered in another coordinator, thus becoming a participant. This is known as “interposition”, meaning there might be a coordinator acting like a participant to another coordinator, so the second one can forward some messages to its participants.

A coordinator can send the following control messages to a participant:

- getStatus: To know the status of the participant.
- AssertionType: To achieve consensus in which will be the format of the protocol messages.
- getIdentity: Which indicates a unique identifier for the participant.

  A participant can basically send messages answering these coordinator messages, and can also send messages to add/remove itself as a participant to a new coordinator and some messages to indicate faulty operations or stats as well.

4.2.1 Comparison with WS-C

There is already another standard that has a similar purpose than WS-CF, the Web-Service Coordinator (WS-C). The main advantages of WS-CF over WS-C are that it is more independent to the implementation of the transaction protocol that is

being used, and also that it's an open standard, while WS-C is not (thus, making it more difficult for application developers to achieve interoperability).

4.3 WS-TXM

The last layer on the WS-CAF is the Web Service – Transaction Management. Its purpose is to allow different kinds of web services transactions, so the system developer doesn't have to care much about the details of the implementation of the transaction protocol.

Mainly, there are three transaction protocols that have been specified to be used in WS-TXM, those are:

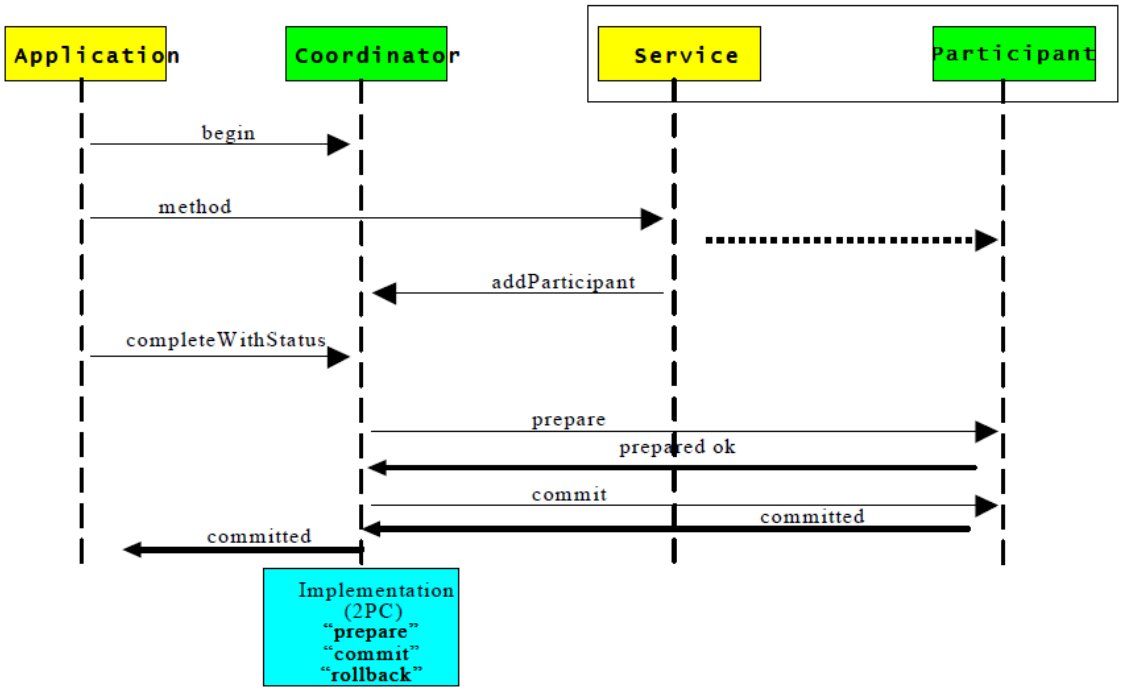
- TX-ACID (ACID Transaction),
- TX-LRA (Long Running Activity)
- and TX-BP (Business Process Transaction).

But WS-TXM is a standard that is prepared to plug in more transaction protocols, so in the future some other transaction protocols may be included in the specification of WS-TXM without much effort.

4.3.1 ACID Transactions

As we've already seen in Shiwei Yu's paper, ACID transactions stands for Atomicity, Consistency, Isolation and Durability. These kinds of transactions are very simple, but they must be taken into consideration when trying to design a framework because of their usage.

A representation of the ACID protocol in the WS-CAF (extracted from [10]):



This diagram works as follows:

1. The Application notifies the coordinator about the beginning of the transaction.
2. Then the application calls the method on the target services.
3. Those services register themselves as participants to the application's coordinator.
4. The application notifies the coordinator about the ending of the transaction, and its status (completed successfully or unsuccessfully)
5. Then the coordinator performs the 2-phase commit protocol with the participants that have been registered (this is, asking for a pre-commit to every participant and if all of them answer positively, then the coordinator tells them to make a permanent commit, if not, it tells them to make a rollback). In case that the transaction hasn't been successfully completed the coordinator tells the participants to rollback the transaction.
6. After that, the coordinator notifies the application about the result of the 2-phase commit.

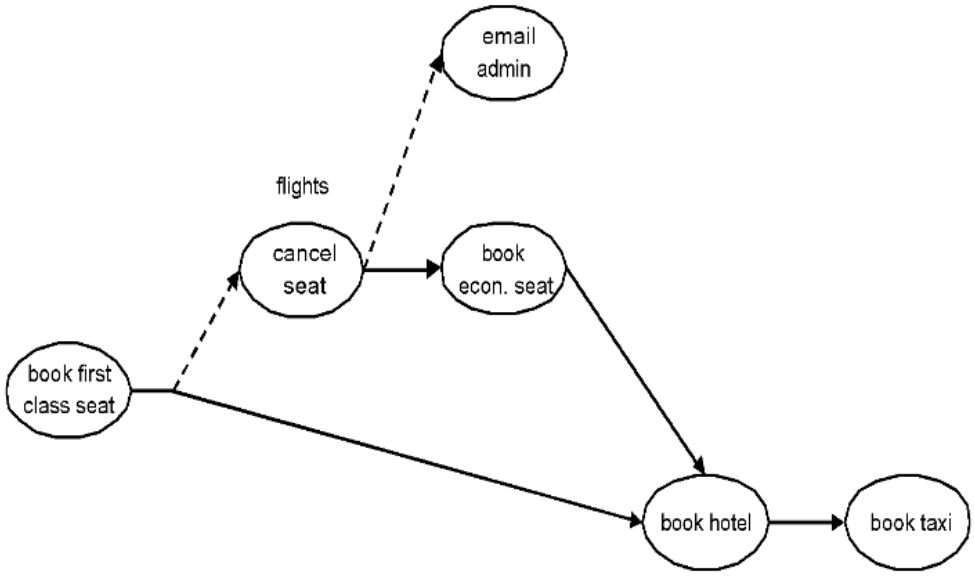
So, in this diagram the application only receives a confirmation of the commitment from the coordinator, thus, doesn't have to perform any of the protocol-specific operations.

4.3.2 LRA Transactions

Long Running Transactions (LRA) are a kind of transactions defined for business interactions that have a large duration. Thus, a LRA transaction can take a lot of time (hours, days, etc) between its beginning and its end. The aim of these kind of transactions is to provide a realistic protocol for actual business processes, for which normal ACID transactions can't perform due that they can't lock resources for such a long time.

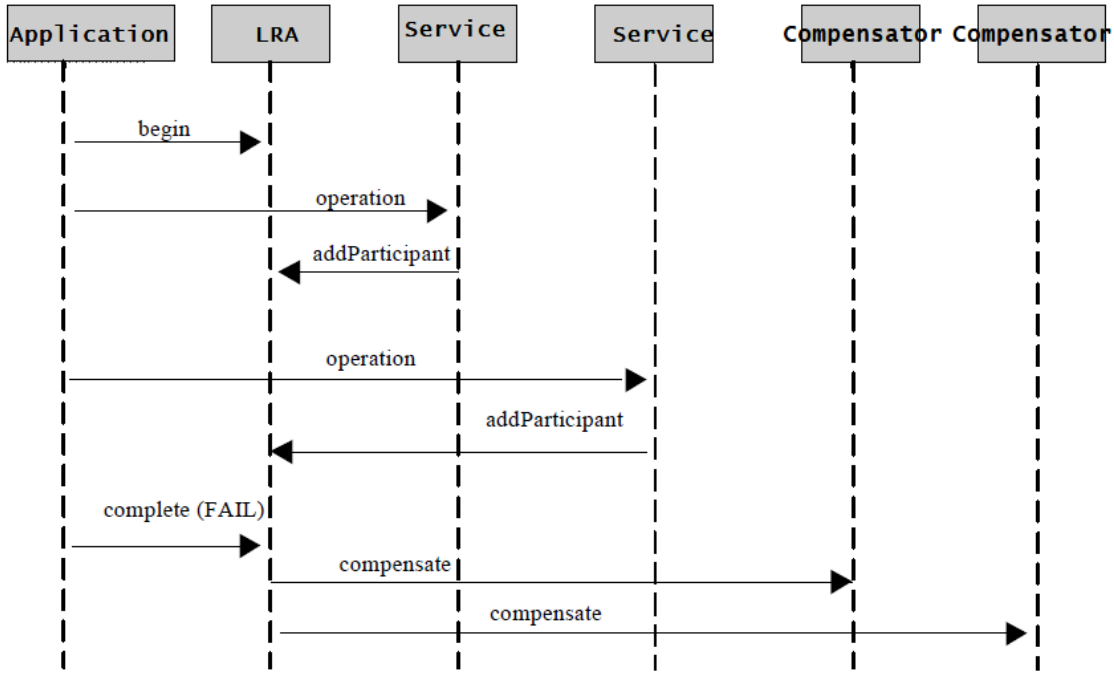
Another of the characteristics of the LRA transactions is the compensability. The LRA protocol is designed to be always compensatable, thus, if it fails in some part and the activity can't be completed properly then another alternative procedure must be done (these alternative procedures might also be compensatable).

An example of a long running transaction (extracted from [10]):



The above picture shows us a LRA diagram, where different tasks are linked by arrows, and compensational tasks are linked by dashed arrows. So, for instance, if we fail in the transaction "book a first class seat" we will trigger the "cancel seat" transaction, which will either succeed or not (triggering another activity depending on it).

An example of how does a LRA transaction works within the WS-CAF (extracted from [10]):



This transaction works as follows:

1. The application notifies the Coordinator about the beginning of the activity.
2. The application performs an operation on the services it needs.
3. Those services are added as participants to the coordinator, providing information about their respective compensators.
4. The application notifies the coordinator about the success or failure of the operation.
5. In case of failure the coordinator activates the compensational procedures.

So, again, the application doesn't need to know anything about the specific implementation of the coordination protocols, since all this work is passed to the coordinator.

4.3.3 BP Transactions

Business Process is a specification designed to make transactions composed of different tasks.

In this kind of transactions there can be:

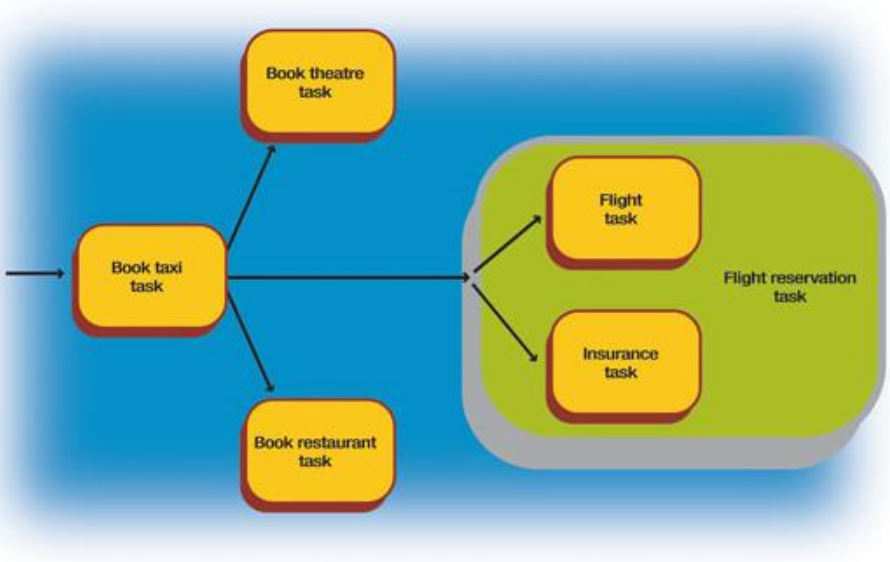
1. ACID transactions
2. Long-running business transaction message

3. Workflow

With the elements mentioned above we can describe a workflow of tasks to be completed. This is, give initial tasks to be completed and link other tasks to be started right when this one has been completed (and so on). Those tasks can also be divided into further sub-tasks. Simple tasks in the workflow can either be simple ACID transactions or long-running business messages.

Thus, the business process allows us to describe business workflows with different kinds of transactions on it, making it a more realistic approach to real processes.

A business process is split into business domains, and each task in the business process belongs to a business domain. A business domain may also be recursively split into other business domains.



The above picture (extracted from [4]) shows us an example of a Business process, where the user firstly books a taxi, and then three other tasks are triggered. One of those tasks (flight reservation task) also has different subtasks. Each of the tasks is in a different business domain.

5 Differences between WS-CAF, WS-T and BTP

WS – Transaction is a standard that works together with the WS – Coordinator mentioned above (in section 4.2.1). One of the disadvantages of the WS – Transaction is that it is not an open standard, while WS-CAF is. Also, WS – Transaction defines two kinds of transaction protocols, the business activity (designed for long-duration activities, a similar idea than LRA) and atomic transactions (implementing an ACID procedure). But the atomic transactions defined in WS-Transaction don't contemplate the two-phase protocol (while WS-CAF does). They only allow simple commit / rollback confirmations between the participants, making it not suitable for ACID transactions that require distributed databases.

Business Transaction Protocol (BTP) also defines two types of transactions, one similar to ACID (atom) and another similar to LRA (cohesion). Atom transactions differ, however, in the sense that some properties, like durability and isolation are not as strictly defined as in ACID transactions. Also, in BTP, the transaction and the coordination are tied, meaning that if any new transaction or coordination protocols are to be introduced it will be more difficult to do it. In the WS – CAF there is a clear separation between the coordination and the transaction (WS – CF and WS – TXM), solving these problems. Finally, BTP enforces the use of a two-phase transaction protocol, while it can't use other commit protocols that WS – CAF can (e.g. simple commit/rollback or three-phase commit protocol).

6 Conclusions

Basically, the variety of protocols used in different services lead the industry to create the WS-CAF standard. Thus, providing a framework to isolate the details of the different protocols used and let the application developers focus on the composite application. The main advantage then is to be able to interact with different service protocols in the same application without the need of knowing the actual implementation of any of those.

This standard is separated into three different standards, the WS-CTX, for managing the context information between the composite application and the different services it uses. The WS-CF, a coordinator framework designed to manage the coordination protocols and make it not necessary for the composite application to know the details of their implementation. It can handle protocols like the two-phase commit (necessary for the ACID transactions). The third standard is the WS-TXM, which implements three transaction protocols (ACID transactions, LRA transactions and BP transactions). This specification is also prepared to include more protocols in the future.

So basically, the application developers don't have to care about which transaction and coordination protocols are used and how the context is propagated / modified, they can focus on which services are used by the composite application.

7 References

- [1]. Bunting, D. and M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenson (2003a). *Web Service Context (WS-CTX)*, July 2003
- [2]. Bunting, D. and M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenson (2003a). *Web Service Coordination Framework (WS-CF)*, July 2003
- [3]. Bunting, D. and M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenson (2003a). *Web Services Transaction Management (WS-TXM)*, July 2003
- [4]. Little, M. and J. Webber, (2003) Introducing WS-CAF – more than just transactions. Web Services Journal, 3(12):52-55, December 2003.
- [5]. M. Little, J. Webber (2003), Introducing WS-Coordination, SOA Magazine [http://soa.sys-con.com/node/39751]
- [6]. OASIS WS-Context specification [http://docs.oasis-open.org/ws-caf/ws-context/v1.0/OS/wsctx.html]
- [7]. OASIS Web Services Composite Application Framework (WS-CAF) Technical Committee [http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf]
- [8]. A. P. Chaudhari**,** B. Majumdar**,** S. Saxena, Long Running Transactions in SOA [http://soa.sys-con.com/node/318438]
- [9]. Bunting, D. and M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenson (2003a). *Web Services Composite Application Framework (WSCAF)*, July 2003.
- [10] Mark Little, Arjuna Technologies, Ltd, "WS-CAF in a Nutshell", WS-CAF face-to-face, Boston 3rd to 4th December 2003.
- [11] Web Services Coordination (WS-Coordination) specification, [http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-tx/WS-Coordination.pdf]