Concurrency Management in Transactional Web Services Coordination

Adnene Guabtni, François Charoy, Claude Godart

Concurrency Management in Transactional Web Services Coordination

Adnene Guabtni[1], François Charoy[2], and Claude Godart[3]

[1] guabtni@loria.fr
[2] charoy@loria.fr
[3] godart@loria.fr
University Henri Poincaré Nancy 1
INRIA - LORIA laboratory, BP 239, F-54506
Vandouvre-lès-Nancy Cedex, France

Abstract. The Business Process Execution Language BPEL4WS has emerged to introduce process dimension in Web Services coordination. At the same time, a lot of needs related to business process management appeared. In this article we focus on transactional management in Web Services platforms. WS-Transaction specification had a big impact on usage of Web Services in critical situations such as financial services. This usage of transactions in web services coordination also introduced concurrency problems similar to those encountered in transactional databases world due to hard transactional constraints especially for isolation mechanisms. Today, WS-Transactions provide flexible atomicity in Web Services coordination (WS-BusinessActivity) but isolation flexibility is not provided. Isolation mechanisms used today are not really adapted to Service Oriented environments and we aim to make them more 'process friendly'. In this paper, we focus on this important part of concurrency problems and propose a new view of WS-Transactions based on Behavioural Spheres approach. This contribution suggests a reorganisation of the WS-Coordination framework adding WS-IsolationSphere for isolation management and the WS-Sphere coordination type for generalising any behaviour management in Web Services coordination.

1 Introduction

Companies are migrating their applications towards a Service oriented Architecture and Web Services represent an adapted way to perform interoperability between different platforms. In such environment, distributed and composed e-services were very useful and specifications like BPEL4WS [6] have been defined to build business processes in Web Services environments. Web Services Orchestration introduced workflow concepts in this context and revealed some transactional needs such as atomicity and isolation to ensure correct execution. These needs were usually performed by traditional transaction protocol such as the two phase commit (2PC). WS-Transactions were introduced to relax isolation in composed Web Services by releasing locks on transaction resources before

their completion in order to permit other transactions to access them concurrently. Implementations of WS-Transaction specification are currently emerging in order to ensure this flexibility in existing platforms but problems related to undesired phenomena caused by this flexibility are not yet resolved. Suppose that a transaction T_1 uses a resource R_1 on which it specified a lock. Suppose also that, due to flexibility reasons, T_1 releases the lock on R_1 before its completion and a transaction T_2 uses the resource R_1 while T_1 has not yet finished its execution. If T_1 fails, all changes made on the resource R_1 should be cancelled (this process is called compensation). Also transactions that depend on data updated by T_1 should be aborted to ensure some coherence of the data/execution. This kind of problem may represent a strategic issue for processes that need a high level of correctness such as financial services.

In this article we propose a point of view that considers isolation as a property that concerns not only activities but also processes. Isolation will be related to sets of business activities allowing adaptation of concurrency during the business process execution. That will enhance concurrency management in Web Services coordination and make it possible to maintain a high degree of flexibility while cascade cancellation and similar undesired phenomena can be reduced considerably. This approach is inspired from sphere of control of C. T. Davies [7] and Isolation spheres that we started to define in [3] and this carries out us to introduce the concept of WS-IsolationSphere. We propose also to reorganise the WS-Coordination framework introducing WS-Sphere coordination type for generalising any behaviour management in Web Services coordination.

We start our paper by presenting what has been done in Web Service Transactions support and what are the different problems related to it. Then we propose our approach based on Isolation Spheres to ensure a process-friendly isolation. Finally we explore the implementation of such approach in the WS-Coordination framework and extensions to other behaviour.

2 Related work : WS-Technologies

Business process specification for Web Services BPEL4WS started in the last five years with an expansion of multiple specifications (XLANG, BPML, WSFL, BPSS, WSCL, WSCI, WS-Choreography and BPEL4WS). These specifications were defined based on numerous business process challenges such as coordinating communication between services, correlating message exchanges between parties, implementing parallel processing of activities, transforming data between partner interactions, supporting long running business transaction and providing consistent exception handling.

Our research focuses on Web Services orchestration/choreography but some clarification needs to be done about the signification of the terms orchestration and choreography. We mean by a Web Services orchestration the execution of a business process under control of a single endpoint (inside organisation, commonly workflow) while choreography represents the observable public exchange of messages, rules and agreements between different business process endpoints

(between organisations). Contrary to orchestration, choreography does not include concurrency management because concurrent data access is usually limited to the same endpoint.

As shown in figure 1, BPEL4WS and CDL4WS (Choreography Definition Language for Web Services) represent the Business Process management both inside and between organisations. Other protocols like WS-Reliability, WS-Security, WS-Coordination and WS-Transactions ensure the Quality of Service part of the entire Web Services Business Process environment.

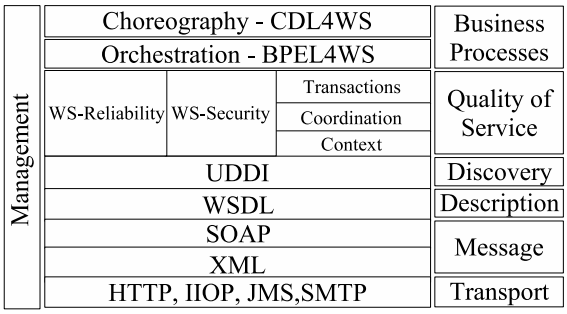


Fig. 1. Standards used with BPEL

In this article we focus on the WS-Coordination and WS-Transaction levels and this will not include WS-Choreography. The challenge of our work is to express flexible isolation requirements in business process environment. First we detail what is already done in WS-Coordination [4] and WS-Transaction specifications [5].

WS-Coordination / WS-Transactions

WS-Coordination specification provides standard mechanisms to create and register services, using separately defined protocols to coordinate the execution of distributed operations in a Web Services environment. It defines a coordination framework supporting the following services :

- Activation Service to create a new coordination activity and to specify the coordination protocols available for the activity.
- Registration Service to register participants and to select a coordination protocol for the activity.
- Coordination Service for activity completion processing using the selected coordination protocol.

WS-Coordination specification proposes customisable coordination types and protocols. WS-Transaction specification represents the specification of two WS-Coordination types that are WS-AtomicTransaction and WS-BusinessActivity as follows :

– WS-AtomicTransaction represents the coordination of activities that express the 'all or nothing' behaviour. Two protocols are possible:
  - Completion protocol : usually, the coordination initiator uses this protocol to tell the coordinator to try a commitment or a rollback.
  - Two phase commit (2PC) : A participant registers to the 2PC protocol in order to initiate a two Phase Commitment with other participants that registered also to the same protocol. Two types of this protocol are available : Volatile 2PC (used for volatile resources such as cache) and Durable 2PC (used for durable resources such as database).
– WS-BusinessActivity handle long lived activities by allowing partial commitment of participants. Business activities support two coordination types
  - AtomicOutcome coordination type must direct all participants to close or all participants to compensate.
  - MixedOutcome coordination type may direct each individual participant to close or compensate.

  In both coordination types, registered participants can choose between two protocols possible with WS-BusinessActivity. These two protocols introduce the notion of completion decision maker as follows :
  - BusinessAgreementWithParticipantCompletion : When a participant registers for this protocol with its coordinator, it must know when it has completed all work for the business activity. The participant must notify its coordinator when its work is done.
  - BusinessAgreementWithCoordinatorCompletion : When a participant registers for this protocol with its coordinator, it relies on its coordinator to tell it when it has received all requests to perform work within the business activity.

3 Motivation

Transactional behaviour supported by the WS-Transaction specification ensures correct and flexible atomicity in Web Services platforms but do not provide such flexibility for isolation constraints. This is due to the fact that isolation in business processes is managed most of the time by the underlying database system and this thanks to locking strategies on data. Some of these strategies provide more flexibility like the SQL isolation levels but they are adapted only to database systems. They also do not take sufficiently into account process aspects because accesses to data are usually considered as independent accesses without particular relations between them.

In such context, we need to provide a 'process friendly' isolation strategy that provides the missing flexibility in transactional Web Services coordination. We propose isolation spheres for Web Services as a solution to the problem and we propose a perspective for similar solutions based on behaviour spheres. In the next section we expose the isolation spheres approach.

4 Isolation Spheres approach

Isolation Spheres are inspired from the spheres of control [7]. An isolation sphere is defined by a set of activities inside a process. For these activities we want to ensure some properties regarding data accessed by the activities (Cohesion property of a sphere) and data produced by the activities (Coherence property of a sphere).

Cohesion property is based on the notion of cohesive view on data. This means that all activities of the sphere have the same view on the data they access which ensures that data values used by the sphere's participants have been set (created or updated) by :

- Activities **participant** to the sphere (any internal data manipulation performed by a participant is visible by the other participants to the sphere). We call such data values 'Sphere-Produced Values' (SPV).
- Activities **non-participant** to the sphere but their manipulation of the data has been performed before any sphere's participant started. We call such data values 'Pre-Sphere Values' (PSV).

We introduce in this paper the notions SPC and PSV to clarify the means of cohesion in such context. These two notions are applicable to data values and not data item itself. This means that a data can have a value that is PSV and after a participant update, it have a new value that is SPV. That's why SPV and PSV data values are disjoint due to the disjunction of participants and non participants to the sphere. Updates done by activities outside of the sphere during the execution of the sphere's participants need to be avoided. This cohesive view represents the basis of cohesion in a group of activities. Cohesion is expressed through different cohesion levels [3] that are Read Uncommitted, Read Committed, Repeatable Read and Serialisable.

These levels define the way the common view of the sphere on data is managed and depend on the nature of used data values. We say that a data value is an '**uncommitted**' one if it is the result of a manipulation performed by an activity not yet completed (usually such data values represent uncompleted or temporary values). If such activity is completed, the data value is called '**committed**'.

- **Read Uncommitted** level allows the participants to the sphere to use uncommitted data during their execution (both for SPV and PSV).
- **Read Committed** level allow the participants to the sphere only reading committed values during their execution (both for SPV and PSV).
- **Repeatable Read** level allows the participants to the sphere to read values of data (both for SPV and PSV) with the certainty that during their use of the data, they will not change.
- **Serialisable** level emulates an execution in series of the activities participating to the sphere. Such execution concerns only the sphere's participants and prevents concurrency inconvenience between them. The execution is similar, from data changes point of view (both for SPV and PSV), to a serial execution of activities of the sphere. External activities do not suffer of such

hard constraint except if they access to data used by the sphere's participants. In this latter case, they are constrained to be serialised with sphere's participant's activities.

Coherence of a sphere represents how activities of the sphere share their data outside of the sphere. In order to control the coherence between data used by activities of the sphere and those by the rest of the processes including concurrent isolation spheres, it is essential to define a level of coherence of the sphere. Isolation spheres ensure some cohesion inside the group and also some coherence of the activities external to the sphere using the same data. The levels of coherence are the following:

- **Cooperative coherence :** All values of data written by the activities participating to the sphere are visible outside of the sphere as soon as they are produced.
- **Activity coherence :** Only values written by the terminated activities participating to the sphere are visible outside of the sphere.
- **Sphere coherence :** The sphere acts as a transaction. The values written by activities participating to the sphere are visible at the end of the execution of all activities of the sphere (that we call also the end of the sphere).

The control of the two dimensions (cohesion + coherence) makes it possible to define in finer way isolation requirements for groups of activities. The choice of cohesion and coherence levels allows estimating the degree of divergence between activities of the sphere and those external to the sphere and the degree of data exchange flexibility between the activities of the sphere and those outside.

5 Isolation Spheres / Behavioural Spheres as Web Services challenges

We consider Web Services platforms as very interesting area for isolation sphere application. The WS-Coordination is able to accommodate the isolation sphere approach. WS-Coordination allows grouping several Web Services that can join or leave the coordination service. This behaviour express what isolation spheres aim to perform : grouping services together over a coordination service and ensuring flexible isolation constraints on their execution.

Although isolation spheres goal concerns transactional behaviour, we think that we need to separate atomicity needs and isolation ones. We propose to include a WS-IsolationSphere specification to the WS-Coordination types family without including it to the WS-Transaction one. To make possible this separation, we propose to reorganise the existing WS-Coordination types. We propose to consider WS-Coordination types as parts of one of the multiple Behavioural Spheres (Atomicity sphere, Isolation Sphere, Compensation Sphere, Multiple Instantiation Sphere, ...).

Behaviour Spheres propose to make separation of concerns between process design and behaviour properties specification. Properties concerned by this approach were related to transactional behaviour and spheres introduced a lot of

flexibility and expressiveness in this topic such as Atomicity Spheres [9][1][10] and Compensation Spheres [8]. We also identified other applications such as instantiation management with multiple instantiation [2]. In this work, we focus on the case of isolation management. Behavioural Spheres consider some properties (atomicity, isolation, compensation, security, and so on) as applied to groups of activities or sub processes. To clarify which situation is adapted to the Behavioural Sphere approach we provides three main principles to respect :

- **Principle 1 :** Separation of concerns between process design and behavioural properties specification.
- **Principle 2 :** Behaviour supported by a sphere and applied to the entire sphere's activities do not produce the same effects when applied to each activity of the sphere separately.
- **Principle 3 :** The use of Behaviour Spheres introduces flexibility and increases expressiveness compared to non-sphere approaches.

We identified a set of Behavioural Spheres adapted to Web Services environment and we propose to use WS-Coordination to provide a framework for Behavioural Spheres specifications that we call WS-Sphere. We propose a non exhaustive family of WS-Sphere coordination types composed of WS-AtomicitySphere, WS-IsolationSphere, WS-JointCompensationSphere and WS-MultipleInstantiationSphere. These four types that we have identified are those we guess compatible with the Behavioural Sphere approach.

The existing WS-Transaction family can be considered as part of the Atomicity Sphere type. Figure 2 illustrates the global organisation of the WS-Sphere integration with WS-Coordination.

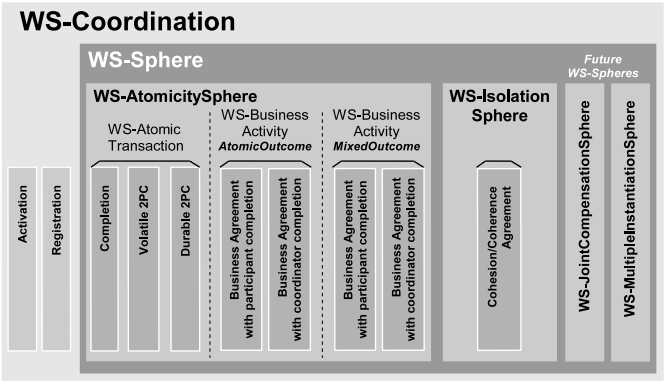


Fig. 2. WS-Sphere integration to WS-Coordination

Our main contribution aims to reorganise the WS-Transaction specification and to propose a solution to isolation management in Web Services coordination.

5.1 WS-IsolationSphere proposal

Our approach based on isolation spheres consider two isolation dimensions : cohesion and coherence. It consists in a process-driven point of view and can also be specified over WS-Coordination and provide a new type of WS-Sphere that we call WS-IsolationSphere.

A WS-IsolationSphere represents a new type of coordination focused on isolation management. It inherits WS-Coordination properties (Activation and Registration) and it is initiated over the choice of a cohesion and a coherence level (Cohesion/Coherence Agreement) when a participant registers to a WS-IsolationSphere. The requirements ensured by the coordination framework depend on these two levels. During the registration process, the participant and the coordinator exchange the coordination context that already implements extensibility elements. We propose to use this standard way to communicate the cohesion and coherence levels to the coordinator registration service similar to the following context :

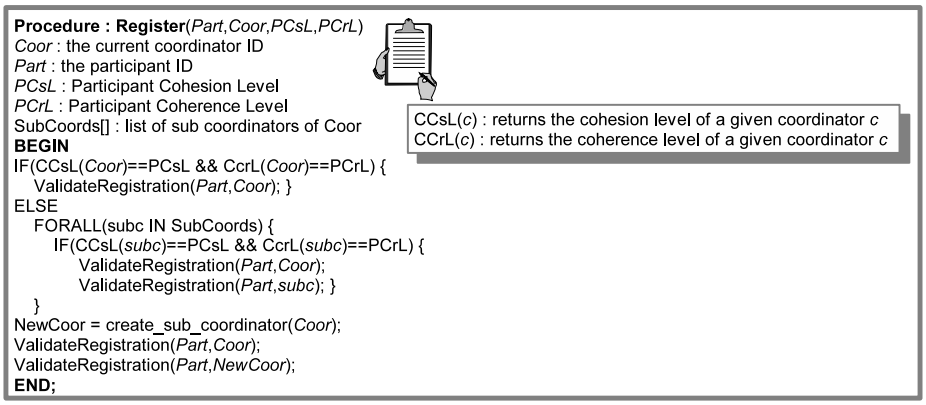
```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
    <S:Header>
        . . .
        <wscoor:CoordinationContext
            xmlns:wsme="http://www.w3.org/2002/06/msgext"
            xmlns:wscoor="http://www.w3.org/2002/06/Coordination"
            xmlns:myApp="http://www.w3.org/2002/06/myApp">
            <wsme:Identifier>http://www.loria.fr/Coor/123</wsme:Identifier>
            <wsme:Expires>2006-10-22T14:30:00.000-05:00</wsme:Expires>
            <wscoor:CoordinationType>
                http://xml-soap.org/2006/03/IsolationSphere
            </wscoor:CoordinationType>
            <wscoor:RegistrationService>
                <Address>
                    http://myservice.com/mycoordinationservice/registration
                </Address>
                <myApp:BetaMark> ... </myApp:BetaMark>
                <myApp:EBDCode> ... </myApp:EBDCode>
            </wscoor:RegistrationService>
            <myApp:CohesionLevel>
                  RepeatableRead
            </myApp:CohesionLevel>
            <myApp:CoherenceLevel>
                  ActivityCohrence
            </myApp:CoherenceLevel>
        </wscoor:CoordinationContext>
        . . .
    </S:Header>
```

The implementation of software capabilities in web services platforms depends on the used information system. We suggest a solution based on middlewares between the Web Service Execution Engine and the DataBase System. It is important to note that isolation management is limited to the same endpoint. Usually, different endpoints do not use the same database and are not located on a same Web Services Execution Engine compliant with WS-IsolationSpheres.

5.2 Nested spheres and Registration/Exit services

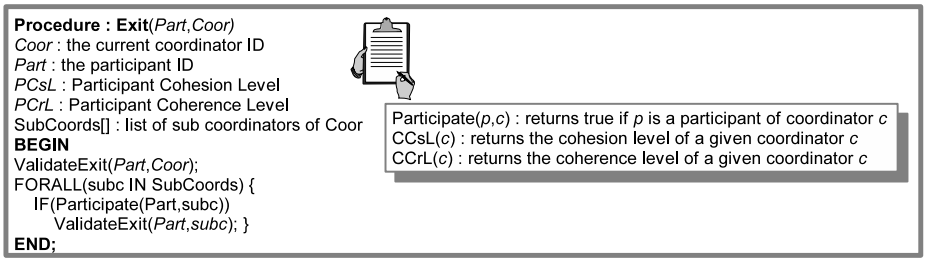
Contrary to WS-Transaction, the registration service for a WS-IsolationSphere is able to decide to redirect a registration to another WS-IsolationSphere. Once an isolation sphere coordinator is initiated by an initiator that determines the coordinator cohesion and coherence levels during the first registration, participants

can register and select both cohesion and coherence levels. Differences between the choice of these levels and those of the coordinator can occur. Also differences from one participant to another can occur. This is a natural phenomenon due to the difference of participant requirements. We propose a procedure executed for each registration to an Isolation Sphere coordinator as follows :



Using the isolation sphere registration procedure, nested spheres are possible. The implementation of such registration procedure enhance the flexibility of isolation management : participants are not obliged to use only coordinator levels but they can propose different levels that coexist with the coordinator's ones.

Isolation spheres can then imbricate and isolation behaviour is enriched. A Sphere can contain other sub spheres that have different isolation needs. A sub sphere defines its own levels for cohesion and coherence but can also benefit from the impact of isolation levels of the upper sphere. Such registration procedure induces an Exit procedure that redirect also to other sub spheres as follows :



6 Conclusion

In this work, we proposed to introduce isolation sphere approach, and more generally behavioural sphere approach in WS-Coordination framework. We integrated WS-Sphere encapsulating WS-AtomicitySphere (including the existing WS-Transaction), WS-IsolationSphere and other proposals such as WS-

ComensationSphere and WS-MultipleInstantiationSphere. We focused on WS-IsolationSphere to introduce flexible isolation management in Web Services coordination using the duality cohesion/coherence levels. We provided registration procedure allowing nested spheres and redirected registration to respond to web services requirements.

More research need to be performed in the following areas. Firstly, we intend to consider compatibility of Behaviour Spheres of different nature and especially atomicity and isolation ones. Additionally we plan to investigate how such heterogeneous Behaviour Spheres can imbricate or even intersect. Finally we intend to implement a full-operational Web Service Execution Engine allowing such WS-Spheres and we aim to implement lower level capabilities to ensure cohesion and coherence levels respect.

References

1. Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willem Jonker. Customized atomicity specification for transactional workflow. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pages 140–147. IEEE Computer Society, 2001.
2. Adnene Guabtni and François Charoy. Multiple instantiation in a dynamic workflow environment. In Anne Persson and Janis Stirna, editors, *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Lavtia*, volume 3084 of *Lectures Notes in Computer Science*, pages 175–188. Springer, Jun 2004.
3. Adnene Guabtni, François Charoy, and Claude Godart. Spheres of isolation: adaptation of isolation levels to transactional workflow. In Wil M.P. van der Aalst et al., editor, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France*, volume 3649 of *Lectures Notes in Computer Science*, pages 458–463. Springer, September 2005.
4. IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, ftp://www6.software.ibm.com/software/developer/library/WS-Coordination.pdf. *Web Services Coordination*, August 2005.
5. IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, http://www-128.ibm.com/developerworks/library/specification/ws-tx/. *Web Services Transaction*, Aug 16 2005.
6. IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/. *Business Process Execution Language for Web Services*, version 1.1 edition, Feb 01 2005.
7. Charles T. Davies Jr. Data processing spheres of control. *IBM Systems Journal 17(2): 179-198*, 1978.
8. Frank Leymann. Supporting business transactions via partial backward recovery in workflow management systems. In *BTW*, pages 51–70, 1995.
9. Frank Leymann and Dieter Roller. *Production Workflow*, chapter Chapter 7 : Workflows and Transactions. Ed. Prentice Hall, Inc., Upper Saddle River, New Jersey, second edition edition, 2000.
10. Willem-Jan van den Heuvel and Sergei Artyshchev. Developing a three-dimensional transaction model for supporting atomicity spheres. *International Workshop on Web Services Research, Standardization, and Deployment*, 2002.