Providing transactional behavior to services coordination

Alberto Portilla

LSR-IMAG Laboratory, BP 72
38402 Saint Martin d'Hères, France
Research Center of Information and Automation Technologies
Universidad de las Américas, Puebla
Sta Catarina Mártir s/n, 72820 San Andrés Cholula, México
e-mail: Alberto.Portilla@imag.fr

Abstract

Our research project proposes an approach for providing transactional behavior to services coordination. In service composition functional aspects have been successfully addressed. However transactional properties have been poorly addressed using *ad-hoc* solutions at the back end of the systems. This situation makes current applications not reliable and not adaptable to existing distributed environments (i.e. Internet). The objective of our research is proposing mechanisms for adding transactional behavior to services coordination.

1 Context and motivation

Service composition is an accepted paradigm for building information systems. This kind of systems are not built from scratch, but using existing software components called services. A *service* is an autonomous software that offers some functionality through a network [1, 2, 7, 25]. Furthermore, services coordination can be used to abstract an application logic by capturing the interactions and dependencies among services. Functional requirements describe the application logic. Several approaches have been proposed to tackle the problem of describing and enacting the services coordination [21, 6, 3].

Let us consider a simplified e-commerce application implementing a purchase process (see Figure 1). Given a purchase order and payment information, it is necessary to get the bank authorization. Once the payment has been authorized, the order is processed, the account is charged, the invoice is sent, and the item is shipped. Notice that there are other issues that must be considered, for instance: an account must be verified before being debited; a canceled order cannot be processed; the invoice process must have been completed when the package is delivered and the account is charged.

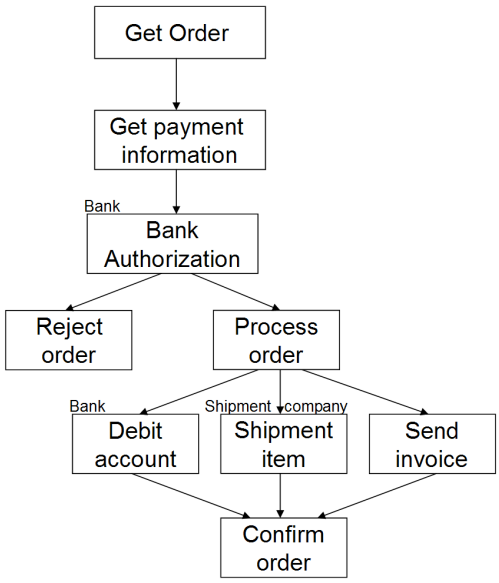

Figure 1: E-commerce application

Actually when an application logic is modeled, application requirements and exceptional situations are also specified. Besides, there are non functional requirements that describe how the execution must be done with respect to some observable attributes like performance, security, transactional behavior, etc. The current approaches for specifying transactional behavior to services coordination can be classified in three groups. Transactional behaviour is specified:

Proceedings of the VLDB2006 Ph.D. Workshop, Seoul, Korea, 2006

1. as a part of the application logic, when it is modeled (e.g. what to do if shipment process fails).

2. as a part of the underlying communication protocol. This approach is adopted by standards such as BTP [14] and WS-Tx [8].

3. as coordination language constructors that are combined with coordination operators (e.g. transaction policies [23], πt-calculus [5], etc.).

Such proposals are mostly based on well known advanced transaction models (e.g. SAGAs, flexible transactions, etc.). Although non functional requirements are independent of functional ones they are in general weaved within the application logic. This situation makes applications very complex in the sense that they are hard to maintain, not flexible and not adaptable [19, 24, 10].

This paper introduces our research project that aims at providing transactional behavior to services coordination. Section 2 states the problem of providing transactional behavior to services coordination. Section 3 discusses the main characteristics of existing solutions. Section 4 sketches the transactional coordination approach that we propose. Finally, Section 5 concludes the paper and gives our research perspectives.

2 Transactional behavior for services composition

Transactional behavior models concurrent access to resources. It is generally related to data. The consistency and reliability of transactional behavior is related to four properties: atomicity, consistency, isolation, and duration [9, 22], known as *ACID properties*. A *classic transaction* (ACID transaction) is a unit of work composed of several database operations that can be committed or aborted usually in milliseconds. Transactional behavior has been tackled successfully in centralized and distributed environments under the support of Database Management Systems (DBMS) [15, 26, 12]. In this environment, a *transaction* is a unit of work composed of several database operations, given a consistent database, it performs actions on it, and generates a new consistent version of the database which is unique and durable [9].

These concepts need to be addressed to services composition under a new perspective and considering the following aspects:

1. Transactions are not related only to data but to execution processes. In the e-commerce example, ensuring the *withdrawal of the account*, *shipment of the item* and that the *invoice sent* are executed atomically concern processes execution and not data.

2. Applications are built by composing loosely coupled services offered by different providers. In our example there are two providers: the bank and the shipment company. These providers do not offer details about services implementation, and in general they do not share data or execution information which are required for executing transactions.

3. The lifetime of processes involved in a service oriented application is expected to be long (i.e. hours, days or weeks). While a classical transaction takes milliseconds to finish, a long transaction takes hours, days or weeks to finish. Long transactions need to be addressed for ensuring the quality of service (QoS) of the application and for avoiding blocking critical resources. Recalling the e-commerce example, *the shipment of the item* can take several days to finish.

Given these characteristics, transactions management strategies must be adapted as follows:

- *Atomicity* must be relaxed to avoid blocking resources for long periods of time, given that the duration of transactions is unpredictable and there are some resources that cannot be blocked during the whole lifetime of a transaction (e.g. bank account).

- *Consistency* cannot be ensured by serial execution when atomicity is relaxed.

- *Isolation* cannot be too strict when multiple transactions operate upon common resources.

- *Durability* to maintain a consistent state across all the transactions executed by the system instead of ensuring persistency.

3 Existing approaches

Several research projects have addressed transactional behavior for information systems, first in the context of DBMS and after for process oriented systems.

In DBMS, transactional behavior has been tackled successfully to data through the concept of ACID transactions [9, 22, 17] and advanced transactional models [15, 12, 26]. These approaches are well suited when data reside in one site and transactions lifetime is short duration. Besides they operate with homogeneous execution units and therefore are not applicable directly to others environments such as Internet.

In workflow systems there are several approaches that aim at ensuring consistency among computations using process as execution units. WAMO [11] introduces a complex transactional language for workflows. [19] introduces how to add atomicity and exception handling for IBM-FlowMark. [18] proposes a workflow definition language to implement the saga model.

[10] addresses atomic behavior to workflows by means of spheres. However these approaches address transactional behavior in an *ad-hoc* fashion and they are in general not implemented.

In Web services, transactions are used to ensure sound interactions among business processes. Web services transactions (WS-Tx) [8] and business transaction protocol (BTP) [14] are the most accepted protocols for coordinating Web services. A coordination protocol is a set of well-defined messages that are exchanged between the participants of a transaction scope. However, these approaches do not offer mechanisms to ensure the correctness in the specification because the developer is on charge of implementing the transactional behavior.

Other approaches provide transactional frameworks. [13] introduces a model for transactional services composition based on an advanced transactional model. [4] proposes an approach that consists of a set of algorithms and rules to assist designers to compose transactional services. These approaches support the definition of atomic behavior based on the termination states of activities. Besides, the execution control flow is defined a priori. This characteristic makes the approaches not flexible.

Transaction policies [23], πt-calculus [5] and transactional Web services orchestration [20] address transactional behavior based on existing protocols (BTP and WS-Tx). Transactional behavior is partially implemented without a clear separation between the application logic and transactional aspects.

In contrast to the above approaches, we consider that transactional aspects can be separated from the application logic and adapted to the characteristics of the services participating in a coordination, and to the application requirements of the target application.

4 Towards a transactional services coordination

In our approach the specification of transactional behavior for services coordination is addressed as follows:

- *Application logic* must be captured by using a coordination approach (i.e. workflow technology).
- *Transactional behavior* must be defined independently of the application logic by using atomicity contracts and associating a well defined behavior to coordination participants. For example the activities *bank authorization* and *process order* of the e-commerce application (see Figure 1) need to be treated as an execution unit that must be atomically executed because according to the application, orders can be processed only if the payment has been authorised by the bank.

We propose an applications development cycle with the following steps (see Figure 2):

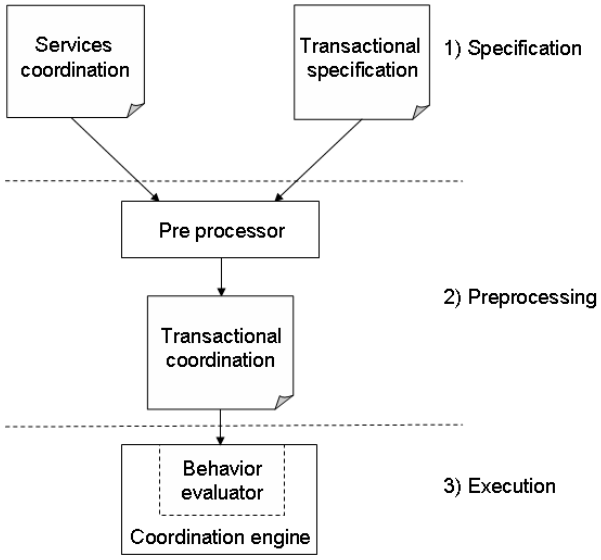

Figure 2: Development cycle of service oriented applications

1. *Specification.* Given a *service coordination* that specifies the dependencies among services and the corresponding exchanged messages a developer can specify the intended transactional behavior (*transactional specification*) according to a model and its associated language that we propose (see below).
2. *Preprocessing.* In this phase, coordination and transactional specifications are weaved by a preprocessor. This process automatically generates a coordination specification and the information required to enact the specification.
3. *Execution.* Transactional coordination is enacted by a transactional coordination engine that has a behavior evaluation module responsible of ensuring the specified transactional requirements.

4.1 Transactional behavior model

We propose a transactional behavior model based on the following concepts:

- *Activity.* It abstracts a service method call. It can be classified according to its behavior on failure [16] by three scenarios: (1) the side effects that can be caused by undoing the activity (e.g. an extra charge for cancelling the activity *debit account*), (2) the possibility or not of undoing an activity (e.g. the activity *ship item* cannot be undone when the order contains underclothes), and (3) the possibility of trying several times an activity (e.g. *send invoice* cannot be retried because the invoice cannot be expedited several times). The behavior of an activity depends on the application semantics. In our example, the activity

bank authorization is vital because no shipment will be authorized without the corresponding payment.

- *Sphere.* It groups a set of activities. It has an associated state, behavior and contract. The notion of sphere is used for modeling atomic properties for execution paths in workflows.

- *Contract.* It specifies the transactional behavior of a sphere and a reaction in case of failure. The behavior is related to well known transactional requirements (e.g. atomicity, isolation, durability, etc.).

The model we propose fulfills the current spirit of reusing practices existing in software engineering. Consequently, it characterizes existing coordination approaches and advanced transactional models. Thereby we ensure that non functional requirements are well abstracted and the resulting specification is sound.

4.2 Execution engine

We have specified the general architecture of an execution engine that consists of two main modules:

- *Behavior evaluator* detects execution failures and generates actions for ensuring the required transactional behavior according to given contracts for a target application.

- *Coordination engine* enacts the coordination specification. We are not interested in proposing a new engine, we define components that can interact with existing engines (i.e. transactional managers, security controllers, etc.).

We are currently conducting the implementation of an execution engine that uses ECA rules for implementing the behaviour evaluator and that can be plugged to a BPEL engine.

5 Conclusion

Emerging paradigms to build information systems, based on services as building blocks, promise to tackle the problems related to distribution through standardization. There are other QoS aspects that need to be addressed to ensure the success of this approach. Transactional behavior has been addressed successfully for managing data and processes execution in centralized and distributed environments. Proposed strategies and models do not necessary apply to services coordination where data is not the unit of transactions, applications are built by using loosely coupled distributed units and transactions are expected to be long duration (hours, days or even weeks). We noticed that existing coordination approaches have addressed non functional aspects in an *ad-hoc* fashion. We also noticed that current transactional services approaches are not flexible.

The main contribution of our approach is a transactional behavior model that clearly promotes a separation of concerns among application logic and the transactional behavior of services. Addressing transactional behavior by, i) implementing a separation of concerns strategy, ii) characterizing execution units, and iii) using recovery strategies, instead of implementing well known transactional models, makes our approach flexible and extensible.

We are currently finalizing the state of the art and detailing our approach. Then, we will propose a framework for specifying transactional behavior managers for services oriented systems. We will also explore how to extend our model to other transactional properties such as isolation and durability.

References

[1] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services - Concepts, Architectures and Applications.* Springer Verlag, first edition, 2004.

[2] Khalid Belhajjame. *Définition et orchestration des services ouverts pour la construction des systèmes d'information répartis.* PhD thesis, Institut National Polytechnique de Grenoble, LSR-IMAG, Juin 2004.

[3] Boualem Benatallah, Quan Z. Sheng, and Marlon Dumas. The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1):40–48, 2003.

[4] Sami Bhiri, Claude Godart, and Olivier Perrin. Reliable web services composition using a transactional approach. In IEEE International, editor, *O. e-Technology, e-Commerce and e-Service*, volume 1 of *eee*, pages 15–21, March 2005.

[5] Laura Bocchi, Paolo Ciancarini, and Davide Rossi. Transactional aspects in semantic based discovery of services. In Jean-Marie Jacquet and Gian Pietro Picco, editors, *COORDINATION*, volume 3454 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2005.

[6] Fabio Casati. eflow: an open, flexible and configurable approach to service composition. In Heiko Ludwig, Yigal Hoffner, Christoph Bussler, and Martin Bichler, editors, *ISDO*, volume 30 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2000.

[7] Christine Collet. The nods project: Networked open database services. In *14th European Conference on Object-Oriented programming (ECOOP-2000)*, June 2000.

[8] William Cox, Felipe Cabrera, George Copeland, Tom Freund, Johannes Klein, Tony Storey, and Satish Thatte. Web services transaction (ws-transaction). Technical specification, BEA Systems, International Business Machines Corporation, Microsoft Corporation, Inc, November 2004.

[9] Claude Delobel and Michel Adiba. *Bases de données et systèmes relationnels.* Dunod, Informatique, 1982.

[10] Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willem Jonker. Customized atomicity specification for transactional workflows. In *Proceedings of the International Symposium on Cooperative Database Systems and Applications*, pages 155–164. IEEE, April 2001.

[11] Johann Eder and Walter Liebhart. The workflow activity model WAMO. In *Conference on Cooperative Information Systems*, pages 87–98, 1995.

[12] Ahmed K. Elmagarmid, Y. Leu, W. Litwin, and Marek Rusinkiewicz. A multidatabase transaction model for interbase. In *Proceedings of the sixteenth international conference on Very large databases*, pages 507–518, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[13] Marie-Christine Fauvet, Helga Duarte, Marlon Dumas, and Boualem Benatallah. Handling transactional properties. In Springer-Verlag LNCS, editor, *WISE 2005: 6th International Conference on Web Information Systems Engineering*, volume 3806, pages 273–289, Octobre 2005.

[14] Peter Furniss. Business transaction protocol. Technical specification, OASIS, November 2004.

[15] Héctor García-Molina and Kenneth Salem. Sagas. In ACM, editor, *9th Int. Conf. on Management of Data, San Francisco, California, USA*, pages 249–259, 1987.

[16] Jim Gray. The transaction concept: Virtues and limitations (invited paper). In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 144–154. IEEE Computer Society, 1981.

[17] Jim Gray and Andreas Reuter. *Transaction processing: concepts and techniques*. Morgan Kaufmann Publishers, 1993.

[18] Paul Grefen, Jochem Vonk, and Peter Apers. Global tansaction support for workflow management systems: from formal specification to practical implementation. *Very Large Data Base Journal*, 10(4):316–333, 2001.

[19] Claus Hagen and Gustavo Alonso. Exception handling in workflow management systems. *IEEE Transactions on Software Engineering*, 26(10):943–958, October 2000.

[20] Peter Hrastnik and Werner Winiwarter. Twso transactional web service orchestrations. *Journal of Digital Information Management*, 4(1):–, 2006.

[21] Amaia Lazcano, Gustavo Alonso, Heiko Schuldt, and Christoph Schuler. The WISE approach to electronic commerce. *International Journal of Computer Systems Science and Engineering*, 15(5), 2000.

[22] M. Tamer Ozsu and Patrick Valduriez. *Principles of distributed database systems*. Prentice Hall, second edition, 1999.

[23] Stefan Tai, Thomas Mikalsen, Eric Wohlstadter, Nirmit Desai, and Isabelle Rouvellou. Transaction policies for service-oriented computing. *Data Knowl. Eng.*, 51(1):59–79, 2004.

[24] Genoveva Vargas-Solar, Luciano García-Banuelos, and Jose-Luis Zechinelli-Martini. Toward aspect oriented services coordination for building modern information systems. In *Encuentro Internacional de Computacion 2003*. ENC-SMCC, IEEE, sep 2003.

[25] W3C. World wide web consortium. Online http://www.w3.org/, April 2005.

[26] Helmut Wachter and Andreas Reuter. The contract model. In Ahmed K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, chapter 7, pages 219–263. Morgan Kaufmann Publishers, 1992.

This research is partially related to the project DELFOS of the Franco-Mexican Laboratory of Informatics (LAFMI) of the French and Mexican governments.

Alberto Portilla is a first year PhD student (double program) at the National Polytechnical Institute of Grenoble (INPG), Laboratory LSR-IMAG, and the Universidad de las Américas (UDLA), CENTIA. He works under the supervision of Prof. PhD. Christine Collet (INPG, France), PhD. Genoveva Vargas-Solar (full time researcher CNRS, France), Prof. PhD. José-Luis Zechinelli-Martini (UDLA, México) and Prof. PhD. Luciano García-Bañuelos (UATx, México). His research focusses on non functional aspects of services based applications. His studies are expected to be finished in August 2008. His studies are supported by the Mexican Education Council through the program for improving the teaching system in Mexican public universities (PROMEP-SEP), the Autonomous University of Tlaxcala, México and the Jenkins Excellence Fellowship Program at UDLA.