International Journal of Advanced Technology in Engineering and Science www.ijates.com

Volume No.03, Special Issue No. 02, February 2015 ISSN (online): 2348 – 7550

REQUIREMENTS, PROPOSALS AND CHALLENGES IN TRANSACTION MANAGEMENT SERVICE-ORIENTED SYSTEMS

[1] Vishal Bhanawase, [2] Sager Mane, [3] Avadhoot Joshi

[1] MTech (CSE),JNTU University, Hyderabad, Andhra Pradesh,(India)
[2] M.E (CSE), Solapur University, Maharashtra, (India)
[3] M.E (CN), Pune University, Maharashtra, (India)

ABSTRACT

Service orientation system has been treated as one the important technologies for designing, implementing deploying large scale service provision software systems. The main idea of SOC is to explicitly separate software Engineering from programming, to emphasize on software Engineering and to de-emphasize on programming. Service-Oriented Computing (SOC) is becoming the mainstream development paradigm of applications over the Internet, taking advantage of remote independent functionalities. The cornerstone of SOC's success lies in the potential advantage of composing services on the fly. When the control over the communication and the elements of the information system is low, developing solid systems is challenging. In particular, developing reliable Web service compositions usually requires the integration of both composition languages, such as the Business Process Execution Language (BPEL), and of coordination protocols, such as WS-Atomic Transaction and WS-Business Activity. Unfortunately, the composition and coordination of Web services currently have separate languages and specifications. A list of potential challenges for the maintenance and reengineering of service-oriented systems is presented for discussion.

Key Terms: Transaction Management, Service Oriented Computing, BPEL

1. INTRODUCTION

Although the Web was initially intended for human use, most experts agree that it will have to evolve probably through the design and deployment of modular services to better support automated use. Services provide higher-level abstractions for organizing applications for large-scale, open environments. Thus, they help us implement and configure software applications in a manner that improves productivity and application quality. Because services are simply a means for building distributed applications, we cannot talk about them without talking about service-based applications specifically, how these applications are built and how services should function together within them. The applications will use services by composing or putting them together. Architecture for service based applications has three main parts: a provider, a consumer, and a registry. Providers publish or announce their services on registries, where consumers find and then invoke them. Standardized Web service technologies are enabling a new

generation of software that relies on external services to accomplish its tasks. The remote services are usually invoked in an asynchronous manner. Single remote operation invocation is not the revolution brought by Service-Oriented Computing (SOC), though. Rather it is the possibility of having programs that perform complex tasks coordinating and reusing many loosely coupled independent services. It is the possibility of having programs manages business processes which span over different organizations, people and information systems. A new approach to software, such as that brought by SOC, calls for new ways of engineering software and for new problems to be solved. The central role of these systems is played by services which are beyond a centralized control and whose functional and, possibly, non-functional properties are discovered at run-time. The key problems are related to the issue of discovering services and deciding how to coordinate them. For instance, while planning to drive to a remote city, one might discover that it is heavily snowing there, and may want to obtain snow tires. Therefore, one needs to find a supplier and a transport service to have the appropriate tires in a specific location by a specific deadline. That is, various independent services are composed into the form of a process, called the 'get winter tires while traveling' with the requirement that we order the tires if and only if we find also a transport service for them. In other words, we require the services of tire ordering and tyre delivery to be composed in a transactional manner. In the present treatment, a service is a standard XML description of an autonomous software entity, it executes in a standalone container, it may have one or more active instantiations, and it is made of possibly many operations that are invoked asynchronously. A service composition is a set of operations belonging to possibly many services, and a partial order relation defining the sequencing in which operations are to be invoked. Such a partial order is adequately represented as a direct graph. A service transaction is a unit of work comprehending two or more operations that need to be invoked according to a specific transaction policy. The coordination of a service transaction is the management of the transaction according to a given policy. One may argue that transaction management is a well-known technique that has been around for ages but, as anticipated by Gray more than fifteen years ago, nested, long-lived transactions demand for different techniques, and in fact they do. To cater for the new features of transactions executed by Web services, various Web transaction specifications have been developed. WS-Coordination specification describes an extensive framework for providing various coordination protocols. The WS-Atomic Transaction and WS-Business Activity specifications are two typical Web transaction protocols. They leverage WS-Coordination by extending it to define specific coordination protocols for transaction processing. The former is developed for simple and short-lived Web transactions, while the latter for complex and long-lived business activities. Finally, the Business Process Execution Language (BPEL) is a process- based composition specification language. In order to develop reliable Web services compositions, one needs the integration of transaction standards with composition language standards such as BPEL. Unfortunately, these are currently separate specifications. This paper has a double goal: The first one is to look at the requirements of transaction management for Service- oriented systems. The systematization of requirements is the starting point for an analysis of current standards and technologies in the field of Web services. The second goal of the paper is to propose a framework for the integration of BPEL with transaction protocols such as WS-Atomic Transaction and WS-Business Activity. We use a simple but representative example across the paper, the drop dead order one, to illustrate

requirements and the proposed approach.

The need for filling the gap regarding transaction management for BPEL in a declarative way is testified also by Other proposals in the Sam line. E.g., independently and in the same time window, Tai et al. have worked out declarative approach to Web service transaction management. Their approach is very similar to ours with respect to the execution framework and the use of a policy-driven approach to extend BPEL definitions with coordination behavior. However, they do not consider the semi-automatic identification of transactions and consequent process restructuring as we do. Earlier, Loecher proposed a framework for a model- based transaction service configuration, though it was never implemented. Even before the birth of Web services, declarative approaches to automate transaction management have been proposed, most notably. The present work extends our survey and requirement analysis for service transactional systems and our proposal of the XSRL language for handling requests against service compositions. In XSRL a construct is defined to express atomicity of services execution, though no means for recovering from failures is provided. The rest of the paper is organized as follows. First, we introduce the drop dead order example.

Fig. 1. The drop dead order example.

Requirements in Section 2 the proposed approach to transaction management is presented in Section 3.

II TRANSACTION REQUIREMENTS

In the field of databases,transactions are required to satisfy the so called ACID properties, that is, the set of operations involved in a transaction should occur atomically, should be consistent, should be isolated from other operations, and their effects should be durable in time. Given the nature of service oriented systems, satisfying these properties is often not possible and, in the end, not necessarily desirable [14]. In fact, some features are unique to service oriented systems: • Long-lived and concurrent transactions, not only traditional transactions which are usually short and sequential.• Distributed over heterogeneous environments.• Greater range of transaction types due to different types of business processes, service types, information types, or product flows. Number of participants.Unpredictable execution length. E.g., information query and flight payment needs 5 minutes; while e-shopping an hour; and a complex business trans- action like contracting may take days. Computation and communication resources may change at run-time.• Unavailability of undo

operations, most often only compensating actions that return the system to a state that is close to the initial state are available. Furthermore transactions may act differently when exposed to certain conditions such as logical expressions, events expressed in deadlines and even errors in case of a faulty Web service. To make sure that the integrity of data is persistent, the two transaction models used are namely Composite and Distributed allow smooth recovery to a previous ”safe” state.

The set of emerging features mentioned earlier, which are a combination of requirements mostly coming from the areas of databases and workflows, provide the basis for identifying the most relevant requirements for transactions in service-oriented systems.

III PROPOSAL FOR INTEGRATING TRANSA-CIONS INTO BPEL

The above survey shows that there are standardized protocols for describing transactions and languages for describing processes in terms of flows of activities. The connection among these is, to say the least, very loose. The problem is that processes are described in terms of activities and roles capable of executing the activities, but semantic dependencies among these activities are not represented beyond message and flow control. It may happen that several operations from a single Web service are invoked within a BPEL process, and dependencies among these operations may exist. For example, before a supplier provides the product requested by a distributor, he needs first to process the request and then reply to the requester. The two operations correspond to two activities in the BPEL process, namely providing products and processing request, which need to be managed in some transactional way, but BPEL is unable to capture the right granularity and the dependencies among operations.

Our proposal consists of making the dependencies among the activities explicit via an automatic procedure and performing a restructuring step of the process, where necessary. The identified dependencies among activities can be then identified by the designer of the process as being transactions or not. In case they are, the designer will decide which kind of transactions they are and simply annotate them. The execution framework then takes care that transaction annotations are correctly managed at run time. Ply annotates them. The execution framework then takes care that transaction annotations are correctly managed at run time.

Let us be more precise on what the phases of the proposed approach are. Consider Figure 3, where data transformation goes from left to right and we distinguish three layers: the data layer at the bottom, the middle execution layer defining the data transformation, and the knowledge level indicating from where the knowledge to transform the data comes. We start with a generic business process designed to solve some business goal. An automatic processing step, which we define next, identifies dependencies among activities. These are then reviewed by an expert that decides which actually transactions are and which not. This step cannot be auto- mated unless further semantic annotations are made on the BPEL. The restructured and annotated process is then ready to be sent for execution. We notice that the restricted process may be sent to execution several times. In fact, at this stage no concrete binding has occurred.

3.1 Preprocessing

Preprocessing the BPEL specification is performed in two steps, namely (I) identification and (ii) resolution of transaction dependencies. In order to illustrate the two steps, we introduce an abstract model of BPEL.1 Abstract model of BBBPPPEEEL specifications A BPEL process specification describes the interaction between services in a specific composite Web service. Its abstract model, known as behavioral interface, defines the behavior of a group of services by specifying constraints on the order of messages to be sent and received from a service [15]. In this sense, a BPEL specification S is a set of activities A and its associated links L, represented by S = (A, L). The links, which are directed, define a partial ordering over the set of activities and are thus well represented as a directed graph (e.g., Figure 4).

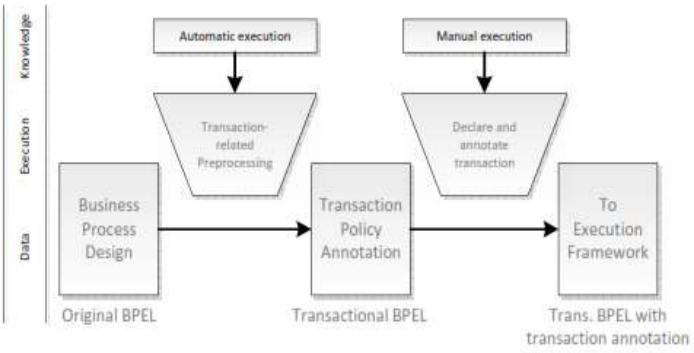


Fig. 3. Approach to integrating transactions into BPEL processes.

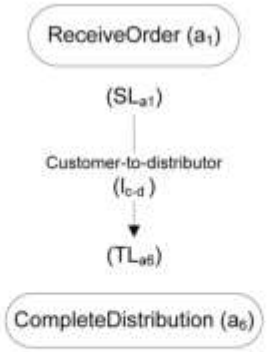


Fig. 4. Representation of activities and the link that connects them.

• An activity a in A having a type represented by

Ta, has the following properties:

- Name Na.
- Operation OPa, which is usually implemented by the Web service at a specific port.
- input variable IVa and output variable OVa, which specifies the parameters required and produced by the OPa, respectively

- set of source links SLa and set of target links T La , which specify the outgoing and incoming links (transitions), respectively.

• A link l in S has a unique name Nl and is indirectly defined through two activities a1 and a2 which indicates not only the direction ld of the transition, but also the conditions lc for the transaction to take place. Furthermore, the Customer-to-distributor link lc−d is one of the source links of the Receive Order activity a1,Furthermore, lc−d∈ La6, where T La6 is the target link of the Complete Distribution

3.2 Dependencies identification algorithm

To identify the existence of transaction dependencies within a given BPEL specification S, we propose Algorithm 4.1. The algorithm is a standard graph algorithm similar to those for reachable set construction, e.g., [16]. The function Identify Dependency takes S as input and outputs a Boolean value that represents the existence of transaction dependencies td. The function first creates a path p for any two activities am and an . Then traverses the links in the link chain ls obtained from p. When a link l is detected and its transition condition lc contains the output variable OVam of the first activity am , or if it contains an output variable OVaI which is identical to OVam semantically, the algorithm stops and returns TRUE. Otherwise, it continues until all pairs of activities in St have been visited. Finally, if no transaction dependencies are detected, the algorithm returns FALSE.

3.3 Declaration of transaction policies

Once transactions are identified and BPEL has been accordingly restructured, one needs to define the desired transactional behavior. One can declare the transaction policy using the following elements:

1) T rams ID is a non-zero integer, representing transactions within a business process.

2) T rams P rotocol specifies a protocol for the transaction, such as WS- Atomic Transaction (WS-AT) or WS-Business Activity (WS-BA).

3) Trans Root indicates the parent transaction identified by TransID. The value 0 is used to indicate the root transaction within the business process.

One can specify the hierarchy of transactions by assigning appropriate TransIDs and Trans Roots.

With such a schema, one can annotate constraints or preferences to a specific activity in the BPEL specification. The annotated activity must be an invoke activity. One can separately specify the desired constraints or preferences in the design-time-info or run-time-info sections. For trans- action management, we declare the transaction policies in the section of the trans-info which is embedded within the section of run-time-info, since a transaction policy is a run-time constraints. Together with the other types of process information, transaction policies are stored in an XML file for use at run-time.

3.4 The Execution framework

The proposed approach transforms a generic business process into a restructured one in which transactions are

identified and annotated. Now one needs an execution framework that is richer than a simple BPEL engine. In fact, one needs to interpret the annotations, make sure that activities are executed according to the transaction conditions and also that the binding among dependent activities is consistent with the transaction semantics. To achieve this we rely on the Sense platform in the context of which the current approach has been developed. Service Centric System Engineering (Sense) is an Europe an sixth framework integrated project, whose primary goal is to create methods, tools and techniques for system integrators and service providers and to support the cost-effective development of service-centric applications [17], [18].

The SeCSE service composition methodology sup- ports the modeling of both the service interaction view and the service process view [19]. A service integrator needs to design both the abstract flow logic and the decision logic of the process-based composition. Therefore, the SeCSE composition language allows the definition of a service composition in terms of a process and some rules that determine its dynamic behavior [20]. Correspondingly, the flow logic can be represented by a BPEL specification, while the decision logic is defined by rules. Based on the architecture of the Sense platform, we built a transaction management tool called DecTM4B. It consists of three modules, namely The Preprocessor for T.M. Is used to identify and eliminate transaction dependencies occurring in the original BPEL specification. The output is the preprocessed BPEL specification.

The SeCSE platform will deal with the binding of abstract services before the BPEL engine executes the BPEL specification. The preprocessing executed by Preprocessor for T.M. happens just before the binding. Currently, ODE and Active BPEL [21] are two BPEL engines supported by the SCENE platform. The Event Adapter maps the low- level events from the BPEL engine onto the binding-related events. The first version of SeCSE event adapter is extended to support the mapping of transaction related events. The Transaction Manager is a separate component in the executor and deployed in the Mule container (Mule is a messaging platform based on ideas from Enterprise Service Bus (ESB) architectures).The Transaction Manager consists of the following two transaction- specific components...

1) TransLog is responsible for managing the lifecycle of transactions, such as creating transaction instances, maintaining the status of transaction instances, and destroying transaction instances.TransLog is also responsible for transferring the information among the components in the executor. For example, it listens the transaction related events from the Event Adapter, and it is responsible for the communication between Transaction Manager and JBoss Transaction Server.

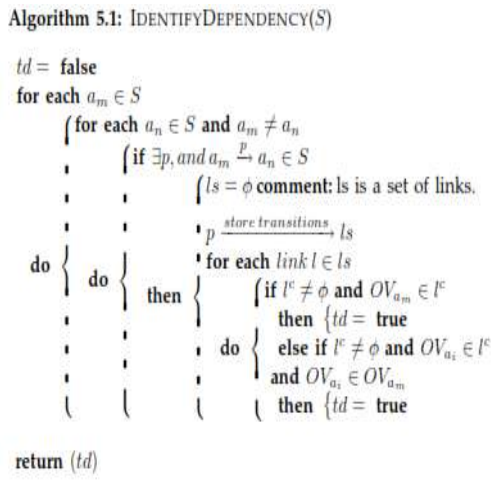
2) Policy Operator retrieves the transaction policies from the XML file, and parses the transaction policies, and then maps transaction policies onto the coordination context. It provides a set of APIs which are to be called by the TransLog.

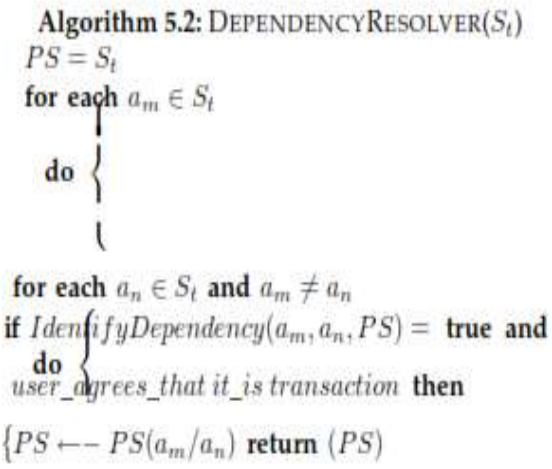
IV RESEARCH CHALLENGES FOR SERVICE ORIENTED SYSTEMS

The following is an attempt to classify research issues in the previously identified domains. The challenges listed under each category are still at a very high level.

They are based on a preliminary literature search, expert opinions from academia and industry, as well as the

author's experience. This list is by no means complete and it is the intent of the authors to gather feedback from a wide community through exposure of the proposed classification and challenges.





V CONCLUSION

The Requirements, Challenges Proposed in our paper consist of business domain and the role of business domain is to focus on activities pertaining to the overall business process as well as on Compliance, trust and analytics. Management. The research pointers in the operations domain focus on activities pertaining to specific application domains, as well as monitoring, support, adoption, and usability. The challenges in the engineering domain focus on activities that relate to the life-cycle of the system from its requirements specification to its maintenance.

REFERENCES

[1] WS-BA, "Web Services Business Activity Framework (WSBusinessActivity), Version 1.1," Aruba Technologies Ltd., BEA Systems, Hitachi Ltd., IBM, IONA Technologies and Microsoft, Tech. Rep., 2007

[2] BPEL, "Business Process Execution Language for Web Services Version 1.1," IBM, Microsoft, BEAT, SAP and Siebel Systems, Tech. Rep., 2003.

[3][Brown06] SOA Governance: How to Oversee Successful Implementation through Proven Best Practices and Methods.IBM white Paper. Ftp: // ftp.software.ibm.com/software/rational/web/whitepapers/10706900_SOA_gov_model_app_v1f.pdf

[4]A. Lazuli, M. Aiello, and M. Papazoglou, "Planning and monitoring the execution of Web service requests," International Journal on Digital Libraries, vol. 6, no. 3, pp. 235–246, 2006.

[5] M. Aiello and A. Lazuli, "Monitoring assertion-based business process," International Journal of Cooperative Information Systems, vol. 15, no. 3, pp. 359–390, 2006.

[6] B. Haugen and T. Fletcher,"Multiparty electronic business transactions. Version 1.1," UN, Tech. Rep., 2002.

[7] M. Little,"Transactions and web services," Communication of the ACM, vol. 46, no. 10, pp. 49–54, 2003.

[8] OASIS, "Business transaction protocol," OASIS, Tech.Rep 2004.

[9] BPEL, "Business Process Execution Language for Web Services Version 1.1," IBM, Microsoft, BEAT, SAP and Siebel Systems, Tech. Rep., 2003.

[10] C. Sun, D. Hammer, G. Biomet, and H. Groefsema, "An evaluation of description and management- standards and languages for Web service transactions," Univ. Of Groningen/ Sense Project, Tech. Rep., 2006.

[11][Gold-Bernstein05] Gold-Bernstein, B. And So, G.Integration and SOA: Concepts,technologies and best Practices.

[12][High05] High, R., Kinder, S., and Graham, S. IBM's SOA Foundation: An Architectural Introduction and Overview. November2005. http://download.boulder.ibm.com/ ibmdl/pub/software/dw/web-services/ws-soa-whitepaper.pdf[13][IBM06]

[13]B.HaugenandT.Fletcher,"Multiparty electronic business transactions.

[14]M.Little, "Transactionsandwebservices,"Communication of the ACM, vol.46, no.10, pp.49–54, 2003.

[15]C.Ouyang,E.Verbeek,W.M.vanderAalst,S.Breutel,M.Dumas, andA.terHofstede,"Formal semantics and analysis of control owinBPEL,"Sci.Comput.Program.,vol.67,pp.162–198,2007. [16]G.Chiola, "Area chability graph construction algorithm based on canonical transition ring count vectors," in Petri Nets and Performance Models, 2001, pp.113–122.

[17] S. Consortium, "http://www.secse-project.eu/," European Union, Tech. Rep., 2005–2007.

[18] The SECSE Team, "Designing and deploying service-centric systems: The SeCSE way." in Service Oriented Computing: a look at the Inside (SOC@Inside'07), 2007.

[19] Various Authors, "Report on methodological approach to designing service compositions(final), version 4.0 SeCSE A3.D3," ESI, CA and CE-FRIEL, Tech. Rep., 2005, http://www.secseproject. Eu/.

[20]"Report on methodological approach to design service compositions (v2.0) SeCSE A3.D3.2.b," CEFRIEL Unisannio, Tech. Rep., 2006, http://www.secse-project.eu/.