Abstract Transaction Construct: Building a Transaction Framework for Contract-Driven, Service-Oriented Business Processes*

Ting Wang, Paul Grefen, and Jochem Vonk

Information Systems Subdepartment,
Department of Technology Management,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{t.wang, p.w.p.j.grefen, j.vonk}@tm.tue.nl

Abstract. Transaction support is vital for reliability of business processes which nowadays can involve dynamically composed services across organizational boundaries. However, no single transaction model is comprehensive enough to accommodate various transactional properties demanded by these processes. Therefore we develop the Business Transaction Framework, which is built on Abstract Transactional Constructs (ATCs). ATCs are abstract types of existing transaction models that can be composed and executed in a service-oriented transaction framework according to the ATC algebra. By selecting and composing ATCs on demand, flexible and reliable process execution is guaranteed.

1 Introduction

With the expanding scale and scope of business process collaboration, the complexity and dynamism of a business process has been dramatically increased. Today's business processes may involve a huge amount of activities, resources and business relationships thus impose a big challenge to compose and manage such processes. Service-Oriented Architecture (SOA) allows distributed applications to be loosely coupled into a cross-organizational business process. E-contracting technology provides an efficient and effective way to ensure trustworthiness between the business parties. Therefore, contract-driven service-oriented processes have been emerging in mission-critical business paradigms (e.g. outsourcing).

Transaction management, which has been widely used in information systems for exception handling and fault tolerance, guarantees reliable and robust process execution. However, the traditional approach of transaction management by locking and afterwards releasing the shared resources per access is not applicable in today's complex and long-lasting processes. Driven by the growing need of reliable service composition and execution in complex business environment, a lot

* The research reported in this paper has been conducted as part of the eXecution of Transactional Contracted Electronic Services (XTC) project (No. 612.063.305) funded by the Dutch Organization for Scientific Research (NWO).

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 434–439, 2006.

of research efforts have been made from both industry and academia. For example, within the CrossFlow project, the X-transaction model for contract-driven enterprise processes in outsourcing paradigms has been proposed[1]. Within the ADAPT project, support for multiple different transaction models for basic and composed services is created [2]. In addition, Web services transaction protocols [3,4] have been proposed by different standardization bodies. These attempts usually address the transaction issues in a very specific application or business environment that are not comprehensive and flexible enough.

According to [5], a solution towards a general transaction support for service-oriented processes is to orchestrate loosely coupled services into a single business transaction by guaranteeing coordinated, predictable outcomes for the participating partners. Following this thought, the XTC (eXecution of Transactional Contracted Electronic Services) project was proposed to develop a Business Transaction Framework (BTF) that provides comprehensive and flexible transaction support for contract-driven, service-oriented business processes. The basic idea of the BTF is to abstract existing transaction models into Abstract Transaction Constructs (ATCs) and compose proper ATCs into a transaction scheme to provide on-demand transaction support. We specify three phases along the BTF life cycle. During the definition phase, the ATC templates are designed. Then during the second composition phase, the needed ATCs are selected to build a transaction scheme according to the process specification. The transaction scheme can be adjusted to accommodate the changes that might take place later on. Last, during the execution phase, real business transactions are instantiated and executed.

In this paper, we introduce and elaborate the novel concept of ATCs, which are the building blocks of the BTF to achieve both comprehensiveness and flexibility. We apply XTraCalm (Cross-organizational Transaction and Contract Algebra and Logic Method) for correct ATC composition and specification. We propose to use e-contracts to specify the Transactional Quality of Service (Tx-QoS) of ATCs. Thus reliable and robust process execution is guaranteed by the ATC-based BTF with Tx-QoS specifications. Due to the page limit, we briefly present our work here and more detailed explanations and examples can be found in [6].

2 Abstract Transaction Construct (ATC) Concept

ATCs are a series of abstract constructs representing the existing transaction models. In fact, they are transactional services that encapsulate transactional semantics and behaviors. ATCs are created during the design phase and the ATC templates exist in the ATC library for later configuration and enactment. We define an ATC as an artifact with the below characteristics:

An ATC has an internal structure. We identify four types of ATC structures: Flat(no internal structure), Sequence, Complex and Tree. A flat ATC is a basic unit representing an ACID transaction. A sequence ATC has the internal structure corresponding to chained or Saga transactions. A complex ATC has the internal structure of the mixed type of arbitrary sequences and parallels, which

roughly corresponds to some complex workflow transaction models. A tree ATC has the internal structure like a tree, which corresponds to nested or similar transaction models. Please note that the tree-like ATCs have the structure of parent-child relationship with no control flow between the nodes. By abstracting existing models with various structures, we mean a semantic abstraction and do not consider their implementation details (e.g. the underlying transaction processing systems).

ATCs are composed in a recursive manner. When viewed from multiple layers, a top-level ATC can be decomposed into several second-level ATCs and the decomposition can go further if necessary. We regard a business process as a single transaction. In service-oriented environment, the component services within a process are regarded as sub-transactions, connecting with each other horizontally or vertically. This way, the whole business transaction can be abstracted as a top-level ATC, while each sub-transaction is abstracted as a second-level ATC and this may go on to get the Nth-level ATCs. For example, Figure 1-b illustrates an ATC 'A', consisting of the second-level ATCs of 'B', 'C', 'D', 'E' and 'F'. 'F' represents a complex process operated by another party, therefore it needs to be assigned as a complex ATC type and may have many levels of decomposition inside. The complexity of the ATC composition within 'F' depends on the transparency level agreed by the two parties. In our example, only one sub-level in 'F' is relevant. This multi-level view of ATC recursion allows a comprehensive transaction scheme that supports complex cross-organizational business processes.

Each ATC guarantees specific transactional qualities. We define these qualities as Transactional Quality of Service (Tx-QoS). In a service-oriented environment, Tx-QoS can be enclosed in the service description files or service agreements. With the unambiguous specification of transactional qualities, process reliability can be enhanced. We define two sets of Tx-QoS: customer-oriented Tx-QoS representing the transactional requirements in a business context, and provider-oriented Tx-QoS representing the system capacity and technical ability from the service provider. With a mapping between these two sets, the service consumers can expect what Tx-QoS can be offered by the service provider thus can better choose services or partners. Meanwhile the service providers can make use of the Tx-QoS specifications for better monitoring and management of their service quality. In Section 4, an algebra and logic method for specifying ATC composition for Tx-QoS specifications is introduced.

An ATC has a parameterizable interface. Like the description file of a service, the interface of each ATC contains the information of the above characteristics. First the names of the parameters in the ATC specifications are defined while the assignment of these parameters take place the next. There are three types of parameters. The first type of parameters specify the internal structure of an ATC e.g. the specific structure type. The second type of parameters specify the composition information of an ATC such as its predecessors, successors and parallels. The third type of parameters specify the transactional qualities

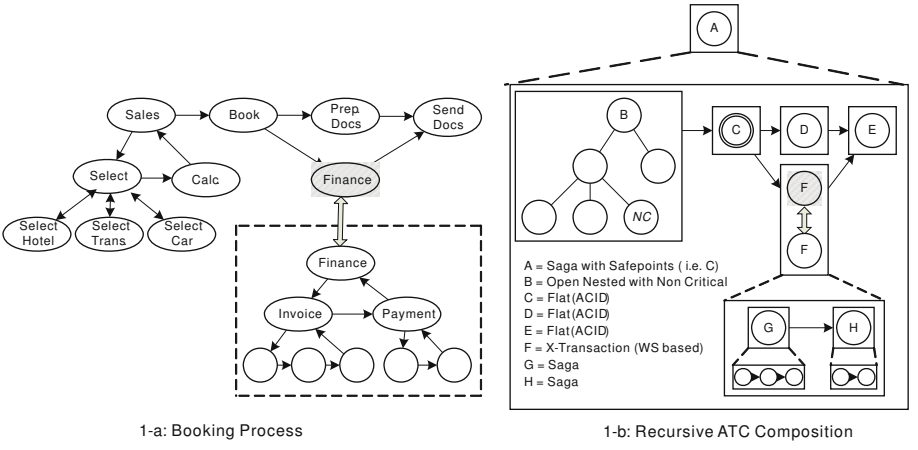

Fig. 1. Example Travel Agency

e.g. atomicity, consistency. When necessary, the composer can assign the recursion level and further refer to other ATCs.

3 Abstract Transaction Construct (ATC) Composition

Upon receiving a process specification, proper ATCs are selected from the ATC library and recursively composed into a top-level ATC for later enactment. To illustrate the ATC concept and especially the composition, we use a variation of the well-known example of a travel agency, shown in Figure 1-a. Customers can create a trip by selecting a hotel, transportation, and an optional rental car (in parallel), after which the costs are calculated and the trip can be booked. Then in parallel the required documents are prepared and the financial issues are dealt with (i.e., invoicing and payment checking), after which the documents are sent to the customer. Being a small travel bureau, the financial dealings are outsourced to a specialized organization, which offers this function through a Web Service. The internal of this Web service consists of invoicing and payment activities, which in turn consist of other activities not relevant for the example.

Assigning certain ATCs, or by redividing the process over ATCs, the transactional behavior will be different depends on the composer's choice upon a particular process specification. As the example process is a long-running one, the entire process might best be supported by a Saga-like transaction model that comprises 'sales', 'book', 'prep. docs', 'finance' (the grayed-out one), and 'send docs'. The selection activities can be supported by an (variation of the) open nested transaction model, as these tasks can be done in parallel. The Web Service needs to be executed under some Web Services transaction model, while the internals of this Web Service, i.e., 'invoice' and 'payment' can be supported by a Saga again. As the Web Service cannot run in isolation, the travel agency might need to see intermediate results when its customers ask for status information, but needs to run in an atomic

fashion so that the available web service transaction protocols (e.g., WS-BA in [4]) do not suffice. In this case, we therefore choose a variation of the X-transaction model [1] that is suitable for the Web Services environment. The resulting ATC composition for this example is shown in Figure 1-b where ATCs are represented by rectangles and the dashed lines represent encapsulation. Eight ATCs are identified and named 'A' through 'H', which correspond to the activities/services shown in Figure 1-a. Note that the unnamed activities that belong to activities 'G' and 'H' are also ATCs but not relevant here.

4 Abstract Transaction Construct (ATC) Formalization

To be able to (automatically) manipulate ATC structures and (automatically) reason about them, we need a more formal approach. In this section, we provide a brief introduction into XTraCalm, a hybrid framework consisting of an algebra and a first order logic. The algebra component of XTraCalm is used to specify ATC structures, the logic component to specify characteristics of (constraints over) ATC structures. As such, XTraCalm forms the basis for Tx-QoS specification in electronic service contracts. For reasons of brevity, this paper only presents a brief glance of XTraCalm and more details can be found in [6].

ATC graphs form the structure of ATC compositions where the XTraCalm algebra is defined. We take A as the domain of ATCs. An element of G is a directed graph of which the nodes are ATCs. Consequently, we can define G as follows: $G = \langle N, E \rangle, \quad N = a \in A, \quad E = \{\langle a, a \rangle \in A \times A\}$. The second level of the ATC structure of the example as shown in Figure 1 can be specified as:$\langle \{B, C, D, E, F\}, \{\langle B, C \rangle, \langle C, D \rangle, \langle C, F \rangle, \langle D, E \rangle, \langle F, E \rangle\} \rangle$.

XTraCalm algebra is used to manipulate elements in G. It is a true algebra in the mathematical sense: it consists of operators that take one or more operands of type G and result type G (it is a closed mathematical system). Currently, we have formally defined operators to combine (both composition resulting forests and concatenation resulting connected graphs) and subtract ATC graphs, as well as operators to extract subgraphs (various forms of chopping and slicing). For example, the *nodes* and *edges* functions result in the nodes and the edges of an ATC graph respectively. The *heads* and *tails* functions result in the sets of ATCs without incoming and outgoing edges respectively. Using the composition ($+$) and concatenation ($\oplus$) operators, we can construct the example graph A introduced above as follows: $\langle B, \emptyset \rangle \oplus \langle C, \emptyset \rangle \oplus (\langle D, \emptyset \rangle + \langle F, \emptyset \rangle) \oplus \langle E, \emptyset \rangle$. Apart from these horizontal operators (operating on graphs at the same aggregation level), we have operators to wrap and unwrap ATC graphs to deal with recursive refinement of ATCs: the wrap operator inserts an ATC graph into a higher-level singleton graph, the unwrap operator results the ATC graph encapsulated in a singleton graph.

XTraCalm logic allows to specify characteristics of ATCs. It is a first-order logic with predicates over A. Using the constructs of the XTraCalm algebra, characteristics over complex ATC structures can be specified. The basis of the logic is formed by base predicates that specify transactional properties of ATCs.

Examples of these base predicates are: $atomic(a)$ that asserts ATC a is strictly atomic and $savepoint(a)$ that asserts ATC a is a savepoint in a saga-like structure. If, for example, we want to specify that in an example graph g, all second 'steps' (in our example case only ATC C) must be atomic and that at least one savepoint must be contained, we can use the following logic expression ($\downarrow$ represents a graph slicing operator): $(\forall a \in g \downarrow 2)(atomic(a)) \wedge (\exists a \in g)(savepoint(a))$. The XTraCalm logic now has a mathematical notation only.

5 Summary and Future Work

To provide flexible and comprehensive transaction support for contract-driven, service-oriented business processes, we have developed the ATC-based BTF. As the building blocks of the BTF, ATCs are encapsulated, parameterizable, composable transactional services, which abstract existing transaction models as reusable constructs. Our main contribution of such a transaction framework lies in three folds. First, it achieves flexibility by selecting and composing ATCs on demand. Second, it uses contractual agreements to specify transactional qualities for processes thereby guaranteeing business trustworthiness. Third, a hybrid transactional algebra and logic for composition and execution is developed to guarantee correctness.

In our future work, the ATC specification will be refined to accommodate even more advanced transactional semantics, which then also requires extending the ATC language. XTraCalm will be extended further to cope with the additional transactional semantics so that reasoning about them, also in compositions with other ATCs, in terms of Tx-QoS is possible. For example, more predicates are needed for full expressions of possible transactional semantics. Moreover, we have to extend the BTF design to specifically cover the cross-organizational aspect that is left out by our present architecture design.

References

1. Vonk, J., Grefen, P.: Cross-organizational transaction support for e-services in virtual enterprises. Distributed and Parallel Databases **14** (2003) 137–172
2. Sorrosal, F.P., no Martínez, M.P., Peris, R.J.: Prototype of the transactional engine, deliverable d4, adapt project (2004) `http://adapt.ls.fi.upm.es/`.
3. Bunting, D., et al.: Web Services Composite Application Framework. available at http://developers.sun.com/techtopics/webservices/wscaf/primer.pdf (2003)
4. Cabrera, L.F., et al.: Web Services Transactions. Available at http://www-128.ibm.com/developerworks/library/specification/ws-tx/ (2005)
5. Papazoglou, M.: Web services and business transactions. World Wide Web: Internet and Web Information Systems **6** (2003) 49–91
6. Wang, T., Vonk, J., Grefen, P.: Analysis on a contract-driven workflow process from a transactional perspective. XTC working document, Eindhoven University of Technology (2006)