A Contract Layered Architecture for Regulating Cross-Organisational Business Processes

Mohsen Rouached, Olivier Perrin, Claude Godart

A Contract Layered Architecture for Regulating Cross-Organisational Business Processes

Mohsen Rouached, Olivier Perrin, and Claude Godart

LORIA-INRIA-UMR 7503
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex,France
{mohsen.rouached,olivier.perrin,claude.godart}@loria.fr

Abstract. As technology infrastructure becomes available for electronic exchange of contracts, the IT community is becoming more interested in modeling of contracts as governance structures for inter-organisational interactions and business processes. This paper investigates e-contract modeling and monitoring. Subsquently, we propose a contract layered model that allows for the convenient monitoring of multi-party contracts during contract fulfillment and reduces complexity of interrelationships. Communication between contract parties rely on a event-based mechanism which extends the scope and flexibility of our model.

Key words:e-contract modeling and analysis, business process management, event-based monitoring.

1 Introduction

Nowadays, there is a renewed interest for modeling and orchestrating cross-organisational and cooperative processes using business contracts. This is motivated by the fact that enterprises increasingly use the Internet for communication with their partners and would like to leverage this technology in order to gain efficiency in contracting processes. Moreover, contracts are important in the context of loosely coupled structures (supply chains for instance). In fact, there is no central authority that coordinates activities of independent entities making up a supply chain, each entity being responsible to arrange a contract with their partner for the collaboration to which they belong.

Usually, contracts define rights and obligations of parties as well as conditions under which they arise and become discharged. The rights and obligations concern either states of the affairs or actions that should be carried out. Often contracts also specify secondary obligations (reparation) that come into force when a party does not carry out an obligation. An e-contract is a contract regulating cross-organisational business processes over the Intenet.

Problems in analysing contracts generally arise from ambiguity and fuzyness of natural langages, the autonomous nature of individual organisations, and the complexity due to the richness of the structures in business organisations. Events that need to be monitored often come from counter parties in other organisations,

and might not be monitorable. Thus, cooperation and trust should be developed among trade partners to alleviate this problem. In general, this improves the transparency of operations, services, and is therefore vital in contemporary e-service providers under strong competitions. It may be the case in SOA (Service Oriented Architecture) or BPM (Business Processes Management).

In this paper, we present a model and a platform to support contracts. We adopt a layered and distributed event-based architecture for modeling and executing electronic contracts. It is worth noting that our approach is completely different from the workflow aspect because we do not specify *how* to manage the business process but we concentrate on regulating cross-organisational business processes over the Internet and determining the responsibility of each partner to respect contract clauses. Our contract approach for coordinating business processes is interesting because it allows the separation of the rules that govern the behaviour of the overall process from the internal processes in the organisations. This feature is important as it allows to ensure the autonomy of the patners, and also to respect their privacy. An other benefit of this apporach is the dynamic adaptation, this means that it is possible to adapt the behaviour of the business process without the need to fully reconfigure it, and changes can be applied without stopping the execution. Then, such an architecture for supporting contract is valuable as it permits to garantee several criteria required for business processes such as expressivity, flexibility, reusability and completeness.

The rest of the paper is organized as follows. In section 2, we present our contract model, while section 3 details contract events. In section 4, related work will be discussed, and section 5 concludes this paper.

2 A Layered Contract Model

To reduce the degree of the complexity and alleviate problems introduced so far, we propose the following contract model, illustrated in figure 1. It is based on a three-layers architecture. This architecture is different from the one proposed by Chiu & al. [CCT02] in the sense that we are not interested in contract negotiation nor contract automatic writing. In fact, we suppose these steps are already done. We are rather interested in the instantiation of the execution infrastructure. Our architecture consists of a *business entities* layer, a *business actions* layer, and a *business rules* layer. Those three layers are coordinated by an event-based interaction mechanism entailing a dispatching and coordination paradigm, which offers the advantage of a complete separation of the coordination aspects and functionality aspects (see section 3). Let us now detail each layer.

The *business entities* layer specifies the organisations involved in the contract. An organisation consists of one or more parties and objects belonging to the parties that are relevant to the e-contract. In the same organisation, each party can participate in several contracts. Thus, our model supports a multi-party contract, avoiding the need to break it down into a number of bilateral contracts. The *business actions* layer captures the details of the actions required in the contract, including the set of roles involved in each action and the set of

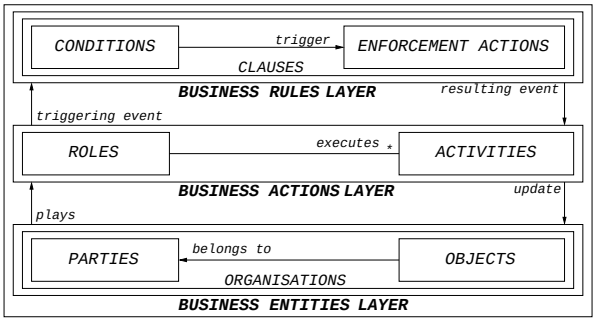

Fig. 1. Contract Model

partners's activities executed by each role. These activities are only those seen from outside during the e-contract execution. We are not interested in internal activities for each party. The *business rules* layer specifies the clauses stipulated in the e-contrat. It consists of two parts: the conditions and the enforcement actions that should be executed under these conditions. The evaluation of the conditions is triggered by a generated event resulting from the execution of an activity in the business actions layer or by an external event.

As such, the layered architecture allows an e-contract to be seamlessly defined and enacted by considering an e-contract as an "abstracted" business process. Then, to map the contract document into electronic format allowing automated management, we carry out two operations, viz., *instantiation* and *execution*. The first operation consists of determining elements of each layer. Then during the execution, the parties start communicating and interacting.

Our approach of sharing the layers between contract parties, illustrated in figure 2, consists of attributing only the business entities layer to each party whereas the other layers are shared between all parties involved. In this figure, a contract is established between three parties $P1$, $P2$, and $P3$. The term *generic* precises that the layer is instantiated by each party whereas the term *inherited* indiquates that the layer is shared by all parties. Thus, each party has its own objects and only those necessary for enacting and enforcing the e-contract are communicated to the others. At the same time all parties collaborate to apply contract clauses and execute necessary activities to accomplish the desired service. As such, we alleviate the problem of using a Third Trusted Party which often could not have only external knowledge about parties which it supervises and thus it could not apply relevant corrective actions. Moreover, this approach permits facilitate update operations for each party. It allows contract parties to be autonomous entities that encapsulate the integrality of their behaviors without any centralized control.

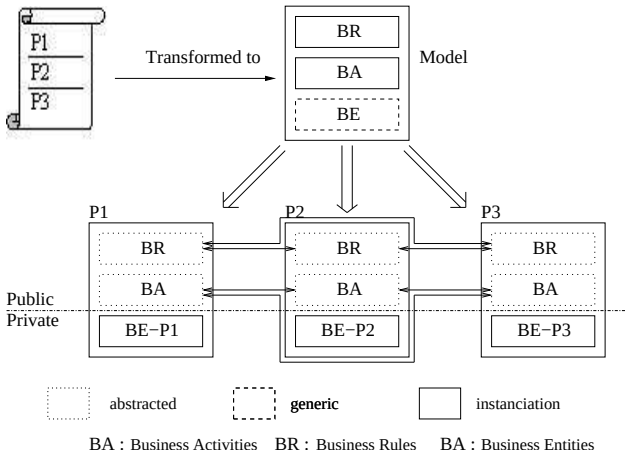

Fig. 2. Global View

3 An event-based architecture to support contracts

Event-based communication is an interesting paradigm for building large-scale distributed systems. It has the advantages of loosely coupling communication partners, being extremely scalable. To synchronize the three levels (Figure 1), we consider an event as a significant occurrence in time or instantaneous (punctual). Events are relevant to roles within a context determined by the contract. This context has attributes which can be *repetition* operators, *detection* mode, *composition* operators, *counting* operators, *negation* operators, and *temporal* management. We have also the possibility to express conditions on these operators.

From an abstract point of view, an e-contract execution can be described by the types and relative order of events occuring in each party. Therefore, after defining events, it is necessary to study their relationships in order to ensure the synchronization of the layered architecture and enable the communication among contract parties.

3.1 Event-driven Causality

Let E_i denote the set of events occurring in a party P_i, and let $E = \cup_{i=1,\ldots,N} E_i$ denote the set of all events in the N e-contract parties. These event sets are evolving dynamically during the e-contract execution. The causality relation $\prec$ onto $E \times E$ is the smallest transitive relation satisfying: (1) if $e_{ij},\ e_{ik} \in E_i$ occur in the same party P_i and $j < k$, then $e_{ij} \prec e_{ik}$, (2) if $s \in E_i$ is a sent event and $r \in E_j$ is the corresponding received event, then $s \prec r$.

Given two events $e1$ and $e2$, if neither $e1 \prec e2$, nor $e2 \prec e1$ holds, they are said to be concurrent.The concurrency relation $\parallel$ onto $E \times E$ is defined as $e1 \parallel e2 \equiv \neg((e1 \prec e2) \vee (e2 \prec e1))$. In general, an unspecified pair of events always satisfies one and only one of the following relations $\forall e1, e2 : e1 \prec e2 \bigoplus e2 \prec e1 \bigoplus e1 \parallel e2$.

3.2 Events contract model

For reliability and efficiency, our event-driven mechanism consists of two meta-models. An **Event Types Meta-model** offers a grammar to describe event types and formal tools to specify composition operators semantics. We specified an event as $(instant, type, validity, cond, mask)$. The *instant* expresses the observation granularity of a special situation. The *type* identifies primitive or composite events defined by applying event operators the primitive ones. The validity interval *validity* indicates the begining and the end moments of the event effect. The *cond* contains information which informs about conditions under which event occurs. The *mask* is a predicate expressing *cond* constraints and temporal expressions that events must satisfy. An **Event Management Meta-model** describes how events are recognized and notified. The context mentioned so far has several properties which include temporal characteristics, semantic characteristics, space characteristics, and state characteristics. Moreover, event relationships are primarily based on concepts of causality and events composition. This proves that the business process automation requires **semantic** level monitoring, rather than **system** level monitoring. Therefore, we focus on relationships between events to deal with monitoring issues, which makes it possible to achieve the pro-active monitoring goal.

4 Related Work

There has been an important numbers of researches concerning the representation of contracts for the purpose of reasoning over, and monitoring, them at run-time. In [Gro99], Grosof introduced a declarative approach to business rules in e-commerce contracts by combining Courteous Logic Program and XML. Marjanovic et Milosevic [MM01] modeled a contract with deontic logic, based on obligation, permission and prohibition. Business Contract Architecture (BCA) [AZAK95] does not provide generic monitoring facilities, expecting each application to develop its own monitoring code to detect and signal non-conformance to the contract monitor. In Seco (Secure electronic contracts) [MK00], the monitoring services allow events to be triggered according to the current state of the contract and informs enforcement service to initiate an enforcement activity. In paper [GLA02], the authors present a three-level process framework for dynamic contract-based service outsourcing and discuss an abstract architecture for dynamic service outsourcing. Comparing with our architecture, this paper did a vertical level research which is involved with workflow system details. On the other hand, our contribution is a horizontal level which interested in interactions among several contractual parties in terms of complex events.

5 Conclusion and Future Work

This paper presents an approach to formalize electronic contracts into a meta-model that enables automatic monitoring. We have detailed a pragmatic archi-

tecture for cross-organisational e-contract enforcement and enactement comprising three layers. We have detailed elements contained in each layer. We have also developed an event-based paradigm to facilitate the executable specification of e-contracting applications.

At the same time, we are working on further details for complex events management and their impact on electronic contracts monitoring. We are also implementing the suggested model using Jena which is a Semantic Web framework containing a reasoner subsystemcd for building Semantic Web applications, allowing both backward and forward chaining.

References

AZAK95. Bond A., Milosevic Z., Berry A., and Raymond K. Supporting business contracts in open distributed systems. In *2nd International Workshop on Services in distributed and Network Environments, (SDNE'95) Whister, Canada*, 1995.

CCT02. Dickson K.W. Chiu, S.C. Cheung, and Sven Till. A three-layer architecture for e-contract enforcement in an e-service environment. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, 2002.

GLA02. P. Grefen, H. Ludwing, and S. Angelov. A framework for e-services: A three-level approach towards process and data management. Technical report, IBM Research Report RC22378, University of Twente, 2002.

Gro99. B. N. Grosof. A declarative approach to business rules in contracts: Courteous logic programs in xml. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC99), USA*, pages 68–77, November 1999.

MK00. Schopp B Greunz M and Stanoevska-Slabeva K. Supporting market transactions through xml contracting containers. In *Proceedings of the Sixth Americas Conference on Information Systems (AMCIS 2000). Long Beach, CA*, 2000.

MM01. O. Marjanovic and Z. Milosevic. Towards formal modeling of e-contracts. In *Fifth IEEE International Enterprise Distributed Object Computing Conference, Seattle, USA*, pages 59–68, September 2001.