Towards integrating business policies with business processes

Zoran Milosevic

CRC for Enterprise Distributed Systems Technology (DSTC)
The University of Queensland, Brisbane, Q 4072, Australia
zoran@dstc.edu.au

Abstract: We present a framework for augmenting business process specifications with policy expressions such as obligations, permissions and prohibitions. One use of such a combined model is to support monitoring of participants' behaviour against agreed policies as in business contracts.

1. Introduction

One limitation of current business process (BP) initiatives, e.g. [1, 2, 3], is their lack of positioning of BP models within a broader enterprise model covering organisational structures, policies and contracts. This paper partly addresses this limitation by applying our community model [4] to a behavioural style typical of BP approaches. The aim of the paper is to augment BPs with policy expressions to support monitoring of participants' behaviour against agreed policies. Section 2 introduces key concepts from ebXML's BP specification (ebBP) [1] and BP modelling notation (BPMN) [3] of relevance for business policies, and illustrate them with a simple example (Fig. 1). Section 3 introduces our policy concepts and maps them onto the relevant BPMN and ebBP concepts. Section 4 discusses open issues and future work.

2. Key collaborative business process concepts

An end-to-end BP model can cover both *private* (internal) and *collaborative* (global) sub-models [3]. We outline key *collaborative* concepts in the ebBP and BPMN standards to show the role of business policies. In ebBP, a *business collaboration* defines a set of roles and a set of business transactions between participants filling these roles. We use BPMN *pool* (shown as a rectangle, Fig. 1) to represent the participants. An ebBP *business transaction* (BT) is a basic unit of work in business collaboration. It specifies how business documents are exchanged between the participants. We use a BPMN *event* to denote the start and end of the collaboration and the arrival of business documents [3]. Each BT has a requesting document flow from the requesting to responding activity and can have a response document flow in the opposite direction. A BT may involve exchange of one or more *business signals* that

support synchronisation of the business states between parties, e.g. to indicate that a document was successfully delivered but cannot be processed by the receiving application because of an invalid document schema. Business signals are separate from lower protocol and transport infrastructure. We use BPMN *message flow* to show ebBP messages and signals (dashed line arrows, Fig.1). A *business transaction activity* (BTA) denotes the use of a BT within the collaboration. The same BT can be performed by multiple BTAs in the same or multiple collaborations. A BTA may specify that a document interchange has a *legal intent* indicating a commitment by the trading partners. The ebBP standard does not provide any recommendation as to how it should be interpreted or enforced, leaving it as a concern of an external contractual framework such as the one proposed in [8]. *Choreography* is the ordering of BTAs, defining the expected flow of business documents and signals and can be shown using UML activity diagrams or other notations like BPMN, as in the ebBP specification.

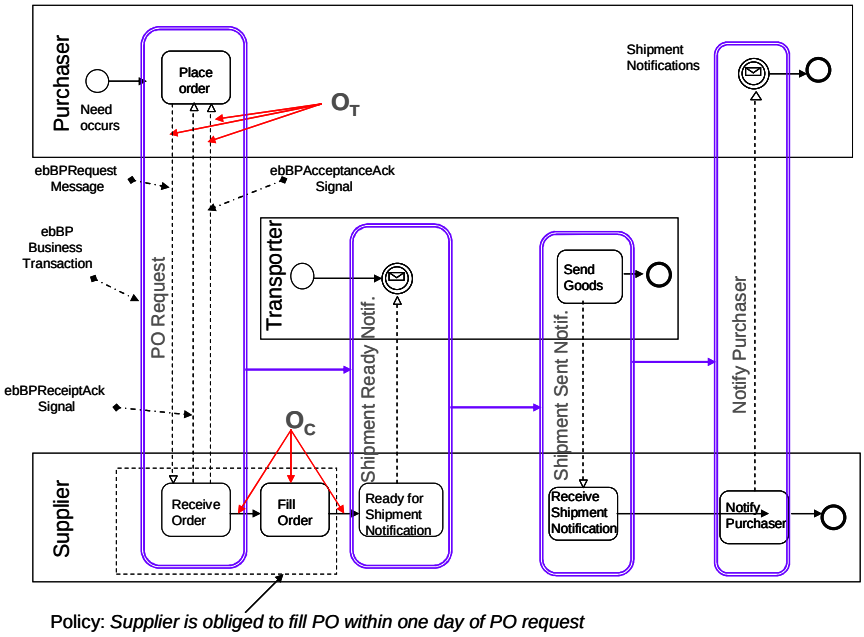

Figure 1: End-to-end business process model

Our example depicts four BTs with their requesting and responding activities. The PlaceOrder activity initiated by the purchaser sends an ebBPRequest message carrying the purchase order (PO) to the supplier. The RecieveOrder activity carried out by the supplier, validates the PO schema and logs this request, if valid, for subsequent processing. This BT involves ReceiptAcknowledgment and AcceptanceAcknowledgment signals. The former acknowledges that the PO is received, while the latter acknowledges PO validity. The signals provide assurance to the purchaser of the successful delivery and acceptance of the PO by the supplier. After the supplier has processed the PO and is ready to supply goods, it sends a Shipment Ready Notification to

the transporter. The BPMN sequence flow shows the ordering between these two BTAs and between subsequent transactions. The figure also shows the FillOrder private activity for the supplier and includes a *business policy*, i.e. *Supplier is obliged to fill the PO within one day of the PO request* (BMPN text annotation). This places an additional constraint on the supplier's FillOrder activity, triggered by successful validation of the PO. The completion of the FillOrder triggers another BTA, between the supplier and transporter.

3. Linking policies and business processes

BPMN and ebBP standards allow the specification of the 'normal' flow of control and data between business activities, i.e. a computationally complete process where points of failure are identified in advance and explicitly represented in the collaboration. Typically, these are network or application failures. In addition, ebBP business signals discussed previously, support the expression of *business failures*. For a successful BT, both its network/application and business aspects must be successful. Although business signals support predictability of interactions when crossing technology and organisational boundaries, they are not sufficient to specify whether business failures are caused by participants not fulfilling their commitments as agreed in the collaboration. Namely, there is a weak link between the specification of business policies that apply to the participants and the 'normal' flows in the collaboration. This is a limitation because a complete specification of the collaboration needs to explicitly state policies that apply to the constituent roles. Consider a situation where a purchaser has received both ReceiptAcknowledgment and AcceptanceAcknowledgment signals. A success of this first BTA only provides guarantees to the purchaser that the supplier is in a position to start executing its own process. However, the purchaser cannot determine whether the supplier has fulfilled their PO and thus satisfied the constraint from the agreement, e.g. 'supplier is obliged to fill the PO within one day of the PO request'. Thus, an added mechanism is needed to deal with unpredictability of behaviour of participants and to check (unintentional or deliberate) violation of their policies. Although ebBP's labelling of BTAs with a 'legal intent' attribute gives special weight to such BTAs, the policy conditions in a collaboration are typically more complex and involve internal actions of parties. In our example, the obligation applies to the supplier (although it is triggered by the purchaser) and it is the monitoring of this policy that determines correctness of their behaviour. In what follows we show how our policy language [4, 8] can be applied to the ebBP concepts to provide a model for the specification of both the normal behaviour and policy constraints.

3.1 Specifying policy constraints

A business *policy* specifies constraints on behaviour of a participant in an organisational context such as an ebBP collaboration. In our unified model [4] this context is called *community*. Community specifies the *roles* involved, their relationships and *basic behaviour* constraints, e.g. control and data flow between participants and/or busi-

ness steps in a BP. A community also supports the expression of policies applying to the roles, e.g. obligation constraints as mentioned before, or a permission constraint such as 'The supplier is permitted to provide an invoice immediately after goods delivery'. A more complex policy expression involving the concept of state is: 'The purchaser has a credit limit with the supplier, which is a maximum outstanding amount with no particular time limit. The purchaser is not permitted to exceed the credit limit' [5]. Basic behavioural constraints and policy constraints are specified using the concept of events and their relationships. An *event* can represent the actions of participants in the collaboration, either their internal actions or their interactions with other parties, or any other occurrence of interest, e.g. the events from the environment or timeouts. Event relationships are called *event patterns* and they have many similarities with the complex event processing ideas [6]. The main role of event patterns is to support event-based monitoring of activities of relevance to policies. An event pattern is evaluated as events come into the system and its progressive evaluation is finished when it's condition is matched. Event patterns range from simple relationships, e.g. sequence of events and logical event relationships to more complex events, e.g. quorum, event causality and temporally-oriented constraints like a sliding time window [4, 8]. Using a simplified version of our policy language, the last policy is:

```
Policy: CreditLimitForPurchaser
        Role: Purchaser
        Modality: Not Permitted
          Condition: PO (OutstandingDebt + PO.value > CreditLimit)
```

Policy is defined in terms of a name, a role to which it applies, modality and event pattern condition (a singleton event of a PO type in this example). Its value is used as a parameter in the condition for checking the value of the *OutstandingDebt* state.

```
State: OutstandingDebt
        CalculationExpression
          UpdateOn: Payment
          UpdateSpecification:
             return (this - Payment.amount)
        CalculationExpression
          UpdateOn: InvoicePurchaser
          UpdateSpecification:
             return (this + InvoicePurchaser.amount)
```

The *OutstandingDebt* state value updates are triggered by the events that affect this state. The concept of state is significant for run-time monitoring of a contract since state variables can be embedded in policy checking expressions as above.

3.2 Applying policies to business collaborations

One motivation for applying policies to the ebBP collaboration is to explicitly associate responsibilities, authorisations, permissions and other policies with collaboration roles which is, for example, needed when integrating contract conditions with the BPs governed by contracts. Another motivation is to support run-time monitoring of participants' behaviour to detect existing or potential violations of the agreed behaviour, as presented in [8] and which is of increasing importance for meeting compli-

ance requirements, e.g. [7]. A further value in separating policies from basic behaviour is the ability to change policies while preserving the fundamental properties of a BP.

The first step in applying policies is to identify events in the collaboration that, via event patterns, are part of policies that apply to collaboration roles, in particular those that are involved in BTs labelled with 'legal intent'. Such events can be part of ebBP transactions, shown as $\mathbf{O_T}$ symbols in Fig.1 (denoting observation points in a transaction), e.g. the sending of Request and Response messages and the generation of ReceiptAcknowledgment and AcceptanceAcknowledgment signals. Each of these can have an associated timeout and their occurrences (e.g. as generated by an ebBP engine) can be modelled as deadline events. These can then be used as input to the business monitoring engine for subsequent management actions, e.g. generation of human readable notifications. The events in a collaboration can, on the other hand, correspond to the transition between ebBP business states in an ebBP choreography. These events are as shown as $\mathbf{O_c}$ symbols (observation points in a choreography). Considering this, a possible implementation of the third policy from section 3.1 is:

Policy PromptOrderFulfillment
Role: Supplier
Modality: Obliged
Condition:
OrderFilled before (ReceiveOrder + 1 days)

The successful receipt of the PO triggers an obligation for the supplier to fulfil the remaining part of the document. The RecieveOrder triggering event can be defined to be: *i)* the AcceptanceAcknowledgment signal (left arrow next to the O_c symbol) or *ii)* state transition at the sequence flow from the RecieveOrder activity to the FillOrder activity (middle arrow next to the O_c symbol). The OrderFilled event (right arrow next to the O_c symbol), if occurring within one day of the RecieveOrder event, signifies the fulfilment of this obligation, otherwise, the violation occurs. In this policy the trigger event was generated by the purchaser and the obligation is on the execution of OrderFilled event by the supplier. Thus, the policy involved the events occurring within two roles in this cross-organisational BP and one is within the scope of the global BP, namely the ebBP transaction (i.e. RecieveOrder event) while the second is within the scope of a private BP (OrderFilled event).

This analysis suggests that our policy language can be used to define additional constraints on the behaviour of trading partners in cross-organisational processes, provided all the events of relevance for the policies are available to the policy specifier. However, the current ebBP standard only allows the specification of the choreography of *collaborative* business activities. This means that the important events in a policy's event pattern that correspond to *private* processes need to have their global counterparts, i.e. these need to be defined as part of ebBP BTAs. An alternative would be to extend ebBP semantics to provide integration points with private processes, or allow for the specification of asynchronous events made visible to trusted third parties for monitoring purposes.

5. Open Issues and Future Work

If policies are to be applied to a cross-organisational BP then the private activities and their integration with the global activities need to be made visible (at least to process designers, not necessarily to end-users), because policy specifications often require the expression of event patterns that include events associated with *either* of the activity types. This is currently not possible in the ebBP models. Further, policy monitoring requires: *i)* the concept of state in ebBP and BPMN and *ii)* definition of a concept of event in ebBP so that ebBP message arrivals, signals and timeouts can be treated as a special kind of that event. BPMN provides a rich set of events and these can be used in ebBP models. Finally, the full power of the community model can further augment cross-organisational BP with other enterprise modelling constructs and provide input to the development of ebBP, BPMN and BPEL standards.

In future we plan to develop a stronger link between ebBP and contract frameworks to enable richer support for contract monitoring, as initially proposed in [9] and contribute to aligning ebXML and legalXML e-contracts [10] standards. We also plan to apply model-driven design for the mapping of policy language to ebBP.

Acknowledgements

The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Education, Science, and Training). The author would also like to thank Andrew Berry and Andy Bond for their comments on an earlier version of this paper.

References

1. ebXML Business Process Specification Schema 3, v2.0 4 Working Draft 10, 23 February 2005 (pre-notification Committee Draft)
2. Business process execution language for web services, May 2003. http://www.ibm.com/developerworks/library/wsbpel/.
3. Business process modelling notation, 2004. http://www.bpmn.org/.
4. P. Linington, Z. Milosevic, J. Cole, S. Gibson, S. Kulkarni, S. Neal, *A unified behavioural model and a contract language for extended enterprise*, Data Knowledge and Engineering Journal, Elsevier Science, October 2004
5. A. Berry, Z. Milosevic, *Extending choreography with business contract*, special issue of the IJCIS journal on contract architecture and languages, to appear.
6. D. Luckham, *The Power of Events*, Addison-Wesley, 2002
7. http://www.sarbanes-oxley.com/
8. Z. Milosevic, S. Gibson, P. F. Linington, J. Cole, S. Kulkarni, *On design and implementation of a contract monitoring facility*, Proc. the 1st IEEE Workshop on e-contracting, July 2004.
9. J. Cole, Z. Milosevic, *Extending Support for Contracts in ebXML*, ITVE workshop, Australian Computer Science Week, Jan 2001.
10. www.oasis-open.org/committees/legalxml-econtracts/charter.php