EFFICIENT: a Tool Set for Supporting the Modelling and Validation of ebXML Transactions

Rik Eshuis
rik.eshuis@tudor.lu
CRP Henri Tudor
29, Avenue John F. Kennedy
L-1855 Luxembourg–Kirchberg
Luxembourg

Pierre Brimont
pierre.brimont@tudor.lu
Bertrand Grégoire
bertrand.gregoire@tudor.lu

Eric Dubois
eric.dubois@tudor.lu
Sophie Ramel
sophie.ramel@tudor.lu

1. INTRODUCTION

The world of B2B commerce is changing today. For more than 25 years, EDI standards, like UN/EDIFACT or ANSI X12, have been the dominant ways of interchanging data between geographically dispersed business applications. These standards are message based: they define the syntax of each message and give some recommendations of use. However, the grammar describing the syntax is often complex and in some cases ambiguous. Hence, introducing new messages and modifying existing ones can be very costly, because parsers and validators that check a complex syntax must be updated. Moreover, rules that constrain the message content and usage are expressed in natural language with the possibility of misinterpretations, resulting in many exceptions that have to be manually handled. This impedes the STP (straight-through-processing) principle which requires a complete automatic management of messages.

To overcome the problems described above, UN/CEFACT and OASIS, supported by several hundred participants, have launched the international ebXML initiative [13]. ebXML stands for ‘Electronic Business using eXtensible Markup Language’. As the name already suggests, a key aspect of ebXML is the adoption of XML as standard language for messages. Besides the message syntax, ebXML has also considered the semantic aspects by recommending usage of a specific UML profile for specifying the message content as well as its business usage. Finally, ebXML has shifted from a message-oriented paradigm into a transaction-oriented paradigm through its associated UMM [4] methodology. This shift results in an end-to-end view of the business transaction where the purpose and the usage rules of each message can be made much more precise than with EDI.

Business experts, even if they are able to read UML syntax, cannot easily validate UML models because these models are usually very complex when they are used to describe real business transactions in which multiple actors participate and in which several messages are exchanged according to complex flows and business rules. Nowadays, validation by business experts can only be done when the whole transaction is implemented and running. Then, business experts are able to discover some missing data elements in messages, some inadequate business rules or even deadlocks or some infinite loops (livelocks) in the transaction flows. Finding such errors will result in a revision of the UML specifications and in the propagation of the new changes in the software
implementation. This process is resource and time consuming, and impedes the rapid adoption and propagation of new B2B communication standards.

This paper describes a tool set, Efficient (E-business Framework For an efficient Capture and Implementation of End-to-end Transactions), we developed for supporting the modelling and validation of ebXML transactions. The tool set consists of a CASE tool and an animator. The CASE tool supporting development of the different UML models associated with the business and analysis representations of the ebXML transaction. The animator allows business experts to validate transactions models at the time they are built, before their implementation has started. The animator is automatically configured from the models produced with the CASE tool. Rather than simulation, we prefer to use the word ‘animation’ since the validation is done in an interactive way, each business expert playing the role of a business actor and participating in the execution of the transaction by receiving messages and sending answers. This way, business experts can validate the transaction by playing different possible scenarios that include different messages. The animator automatically checks the business rules defined on these messages and gives feedback in case of an error.

Validating a specification by animating it has been proven a useful technique in different contexts [1, 2, 3, 5, 11]. Most animation approaches [1, 2, 5, 11] offer non-distributed, centralised animation facilities and are not internet based. Consequently, only a single actor is needed to animate the specification. A notable exception is the work by Heymans and Dubois [3], in which a distributed internet-based animator is presented that allows different actors to cooperatively animate a specification. However, that animator requires a human scheduler for coordinating the behaviour of the different actors, whereas our approach uses a workflow engine for this. Hence, we have automatic coordination whereas Heymans and Dubois require manual coordination. Another difference of our approach compared to the approaches mentioned above is that our implementation is based on open XML-based standards and related technologies, and does not use a proprietary language.

The remainder of this paper is structured as follows. Section 2 explains how ebXML transactions are modelled. Section 3 discusses the architecture of the animator, which is based on a workflow engine. We end with conclusions.

2. MODELLING EBXML TRANSACTIONS

Together with ebXML, a methodological approach, called UMM (UN/CEFACT Modeling Methodology), for the design of business transactions has been introduced. UMM has been developed by UN/CEFACT’s Techniques and Methodologies Working Group (TMWG) since 1998. However, a UMM version [4, 12] was merged with the RosettaNet methodology [10] during the ebXML initiative. Like other UML-based methodologies, UMM relies upon a number of UML models to be developed at different stages of the B2B project. Those models have some specific characteristics, therefore we can consider UMM models as a specific UML profile.

We have adopted a three layered approach (see Figure 1):

- The business layer gives a general overview of the business transactions. The global structure of each business transaction is depicted with a UML use case diagram and a UML class diagram. The use case diagram

Figure 1: Different levels within B2B project

specifies the global structure of the business processes underlying the business transaction. The global class diagram specifies the information manipulated in the business processes. At this layer, there is no concept of message.

- The specification layer details a message-based structure of a business transaction. This detailed specification is needed to support the B2B automation of business transactions. The business process activities of the business transaction are specified using a UML activity diagram. Each message is specified using a UML class diagram. The activity diagram refines the use case diagram at the business layer. Each class diagram is a particular view of the global class diagram at the business layer.

- At the technical layer, the business transaction is executed using the animator. The infrastructure used at the technical layer is automatically configured from the models developed at the specification layer.

In ebXML, business process models have three levels of detail. Swimlanes and object flow states only occur in the lowest level activity diagram, in which the behaviour of two parties that exchange business documents is modelled. To make the specification more easy to comprehend, we decided to merge all these levels into one activity diagram. We can easily transform our activity diagrams into a layered ebXML model and vice versa, because we require that each object flow state has one input and one output activity, which belong to two different swimlanes.

To support the validation of ebXML transactions, we have extended an existing commercial UML-based CASE tool, MagicDraw [9], with plugins that execute the code generation depicted in Fig. 1. Since MagicDraw is a state of the art CASE tool, we expect that the developed plugins can be easily transferred to other UML-based CASE tools.

Because of space limitations, we only give an example of models at the specification layer. The example is about Mercata, an online broker of standard products (taken from the MIT process handbook [7]). We focus here on one particular business transaction: a business client creates an order by submitting his demand for certain products to Mercata. After Mercata provided some technical information on the delivery (e.g. delay, cost), the client can pay. Depending upon the payment means, the payment is checked by either the Credit Company or by Mercata’s accountant. Figure 2 shows an activity diagram of the business transaction. Each
object flow state in this activity diagram models a message, whose structure is modelled with a class diagram. Figure 3 shows the class diagram for object flow state credit_ask. This class diagram is a view of the global class diagram at the business layer. Message class diagrams can be annotated with business rules. These rules are expressed in structured English. We have defined a translation from structured English expression into xlinkit constraints, and we are currently implementing it in a tool. xlinkit [8] is a tool which checks the consistency of xlinkit constraints (XPath-like boolean expressions) against a set of XML documents.

3. ANIMATING EBXML TRANSACTIONS

Once the models have been designed at the specification layer, business experts can interactively ‘play’ with them through the animator. The animator is internet based, allowing business experts dispersed over several places to cooperatively animate the specification through a simple web browser. Each of the experts is responsible for animating one or more of the actors involved in the transaction.

Heart of the animator is a workflow (WF) engine that has been developed before at our research center. In general, a WF engine coordinates the execution of automated business processes. In this case, it coordinates the execution of the business transaction modelled at the specification layer. The business activities of the business process are executed by actors outside the workflow system (i.e. the business experts), not by the WF engine itself. To coordinate the business process, the WF engine needs to have a description of the process. Our WF engine reads process descriptions in XPDL format. XPDL [14] (‘XML Process Definition Language’) is a standard language developed by the Workflow Management Coalition (WFMC). The XPDL process description used during animation is derived automatically from the activity diagram at the specification layer.

Figure 4 shows the overall architecture of the animator. The architecture heavily relies on XML technology. The animator exchanges XML messages with clients using the SOAP protocol. To allow for reuse, the animator architecture consists of many small independent modules. We now discuss the architecture in more detail. First, the XMLMessageHandler module receives XML messages from a client through a SOAP server. The module first stores the XML message in an XML database, eXist [6], and also stores some relevant information, i.e. the name of the recipient, initial sender and the reception time. This information is used in the subsequent checks. Next, XMLMessageHandler calls the WF engine using an API definition defined by the WFMC. Each message signals the end of some running process: the XMLMessageHandler calls the WF engine to terminate this activity.

Next, the WF engine calls the XMLSchemaChecker module, which checks whether the message conforms to its defin-
ing W3C XML Schema. This XML Schema has been derived automatically from the class diagram of the message at the specification layer. In case of an error, an error message is generated and sent back to the initial sender, and the WF engine enables the sending activity again. Otherwise, the WF engine proceeds with the next activity.

If the message is error free, the WF engine calls module XlinkitConstraintChecker, which checks the constraints annotated on the message. These constraints are defined at the message class diagrams at the specification layer. The module calls xlinkit [8], a tool that verifies the consistency of the rules against a set of XML documents, in this case XML messages. The current message is always part of the set. If a rule refers to other (previous) messages, xlinkit retrieves these relevant messages from the XML database. Again, if the verification fails, an error message with feedback is sent to the message sender and the WF engine enables the sending activity again.

Finally, the workflow engine calls module XMLMessageSender. This module computes the message that the recipient of the validated input message could send, in the next activity of the business transaction. The module pre-fills as many fields of the response message as possible with information already known from previous messages. This feature is helpful if the next message should include a lot of information items already included in the current message. However, the recipient can overwrite a prefilled field. The prefilling is governed by building rules, which identify which parts of the current message equal parts of previously sent messages. The next message is attached to the current message, and forwarded to the SOAP server at the recipient site. For example, if message order arrives, XMLMessageHandler finds out that the next activity will output message summary, and it will therefore annotate order with a summary message that is prefilled as much as possible. If the activity started next is a \texttt{<<HumanChoice>>} activity, more than one response message is possible. Note that the recipient, in the next activity, in most of the cases merely has to select the right answer. Hence, prefilling can reduce the workload of the message’s recipient considerably.

4. CONCLUSION AND FURTHER WORK

We have introduced a tool set that supports the development and validation of B2B transactions. The tool set consists of an extension of a CASE tool and an animator. The animator is automatically configured from the models specified in the CASE tool, thereby enabling a fast animation. Moreover, the animator architecture is internet based, allowing business experts dispersed over different places to cooperatively animate distributed business transactions using a simple web browser. Business rules defined in the UML models are checked automatically by the animator. The animator is tightly integrated with state of the art XML technology, showing the feasibility of our approach for ebXML.

Future work includes lifting the business rules from xlinkit to a higher level language. We are currently investigating the use of structured English as a rule language. Another extension of our work concerns activity diagrams. We plan to add the specification of real-time constraints, e.g. timeouts. At the tool level, we plan to interface the CASE tool with a model checker, to allow an exhaustive verification of the dynamic behavior of the business transaction. Error traces returned by the model checker can serve as input for the animator.

Acknowledgements

The research reported upon in this paper has been supported by the FNR under grant FNR/01/01/07 (Efficient). The work of Rik Eshuis has been supported by a grant from FNR and has been performed within the scope of the LIA-SIT (Luxembourg International Advanced Studies in Information Technologies) Institute.

5. REFERENCES