Efficient: A Toolset for Building Trusted B2B Transactions

Amel Mammar[1], Sophie Ramel[2], Bertrand Grégoire[2],
Michael Schmitt[2], and Nicolas Guelfi[1]

[1] Software Engineering Competence Center (SE2C), University of Luxembourg,
6, rue Richard Coudenhove-Kalergi L-1359 Luxembourg-Kirchberg, Luxembourg
{amel.mammar, nicolas.guelfi}@uni.lu
[2] Centre d'Innovation par les Technologies de l'Information (CITI),
Centre de Recherche Public Henri Tudor,
29, Avenue John F. Kennedy L-1855 Luxembourg-Kirchberg, Luxembourg
{sophie.ramel, bertrand.gregoire, michael.schmitt}@tudor.lu

Abstract. The paper introduces an approach to the specification, the verification and the validation of B2B transactions. Based on the usage of a subset of formally defined UML diagrams complemented with business rules, we introduce two facilities offered by the supporting Efficient toolset, namely the checking of formal properties expected from the produced models as well as the animation tool allowing business experts to understand and 'play' with business transactions models before they are implemented. The overall approach is illustrated through the experiences gained in the performance of a real transactional Import/Export business case.

1 Introduction

Message-based Electronic Data Interchange as specified by industry standards such as EDIFACT, EDIFICE, ANSI X12 or VDA is progressively being replaced by transaction-based and extensible specifications covering the needs of the whole business process. XCBL [23], ebXML [3] and Rosettanet PIPs [14] are a few examples of new recommendations that illustrate this trend. These initiatives are defining a set of harmonized business scenarios along with the respective rules, messages and technical requirements that facilitate their implementation. Trading partners that are planning to set up an E-Commerce transaction can develop their own scenario by selecting and adapting an existing standard in accordance with their business needs. The basic building blocks of an E-Commerce scenario are the flow and the rules according to which business documents (messages) are exchanged among the actors participating in the transaction. Together with this new business transaction perspective is also adopted XML[6] and a set of associated non-proprietary technologies as the new syntax for representing the exchanged messages.

Regarding the design of the B2B transactions new recommendations like those included in the UMM method [17] proposes the use of UML[18] in order to produce a set of precise models of the transaction. However, as already experienced by many

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 430 – 445, 2005.

practitioners, some UML diagrams are not enough precise to have a unique interpretation. Furthermore all the semantics of a B2B transaction cannot be captured in terms of UML diagrams that need to be complemented with additional descriptions expressed in natural language. To overcome these problems we have developed a specific CASE tool supporting the design of fully formal UML based models. Associated with these models, and taking direct profit of the absence of ambiguities in them, we have developed two additional facilities, which allow to discover errors in these models, namely

- A verification tool which, based on a proof system, allows to check the consistency and the completeness of the UML models.
- A validation tool which, based on an animator system, allows business experts to understand and 'play' business transactions before they are implemented.

These two facilities allow to discover errors in the UML models at the design time rather than at the implementation and test time. The direct consequence is not only a gain of resources due to an early discovery of errors already at the design time of the transaction, but also an enhancement of trust from business experts into the produced models.

The Efficient approach differs from related work ([26], [27]) in that the business requirements formulated and modeled by a group of business experts are first validated automatically to check that they contain no logic errors, and then translated into an animation that permits the business experts themselves to validate that the transaction models correspond to their business needs.

The rest of the paper is structured as follows. In Section 2, we present the overall approach supported by the Efficient CASE tool. In Section 3, we then focus on the specification layer and illustrate its purpose by means of a case study illustrating a real B2B Import/Export business case. Section 4 details the business rules language that is used in complement to the UML models. Finally, verification and validation facilities are described in Section 5. Section 6 wraps up with conclusions and future directions.

2 Overview of the Efficient Process

Underlying the Efficient CASE tool is a proposed methodology, which makes three layers of descriptions distinct, as illustrated in Fig. 1:

- The *business layer* provides a top-level view on the business scenario that governs the transaction. It depicts the transaction configuration and allows the trading partners to develop a common understanding of the business goals, of the roles and responsibilities that apply, of the various business activities it involves and of each actor's contributions and benefits. In sum, this layer addresses the question of who exchanges what with whom and expects what in return. The main actors along with their respective activities are modeled by a UML use case diagram associated with scenarios capturing the transaction. Furthermore, a global UML class diagram captures the core business concepts and their relationships that are at the heart of the information exchange. We refer to this class diagram as the *business domain* underlying the transaction.

- The *specification layer* details the flow and the business documents (messages) that the trading partners exchange in the transaction. We use UML activity diagrams to model the flow of the transaction and UML class diagrams to specify the content of a business document or message. Each document regroups a subset of the elements of the business domain depending on the information needs related to the business activity the recipient of a message will need to act upon. Each message may be further annotated with a set of business rules (expressed both in an informal and formal way).
- At the *technical layer*, after the transaction has been formally verified, the business transaction is executed using a workflow engine. The infrastructure that this layer is based upon (workflow engine, interfaces, XML messages, rules checker) is automatically configured and generated on the basis of the models developed in the other two layers. The architecture of the Efficient tool set is distributed and Internet-based, allowing trading partners from different sites to participate in the execution of the transactions using a Web browser interface.

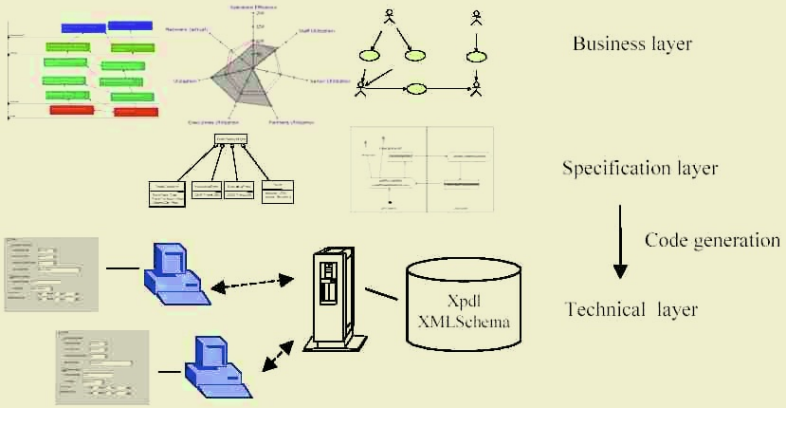

Fig. 1. The modeling layers of the Efficient tool set

More information about the business layer can be found in [16]. Details about the technical layer and more specifically about automated generation of code from UML models are available in [4]. The rest of this paper focuses on the specification layer with a specific focus on the proposed business rules.

3 The Specification Layer

The following sections presents the specification of the static and dynamic aspects of an e-business transaction in the form of UML diagrams, which requires a joint effort of both the IT experts who understand the requirements of the formal modeling languages and the business experts who help to formulate the requirements the transaction must meet. To create these models, the Efficient framework uses a

commercial UML CASE tool called MagicDraw [11] to which a plug-in has been developed that caters for the generation and validation activities discussed in the next sections.

In the rest of the paper we illustrate our ideas on a real business case that we have performed within the context of the EU "EA2" project (Euro-Asian EDI Integration), whose goal is to help the country of Vietnam with the appropriation of international EDI standards to the particular needs of their socio-economic context. The transaction we selected in this case study is based on a European initiative to standardize data exchange between customs (New Computerized Transit System). It defines the various data exchanges required in an Import/Export transaction.

3.1 Business Domain

To begin with, the actors implied in the global transaction must be identified, as well the different associated transactions. In the NCTS import/export transaction, the following roles have been identified:

- The Principal Trader: the party that sends the goods
- The Destination Trader: the party that receives the goods
- The office of Departure: the customs office in the country of departure
- The office of Destination: the customs office in the country of destination
- The office of Transit: the customs office in a country where the consignment is supposed to cross the external border(s)
- The office of Guarantee: the office in the country of departure that registers and maintains a guarantee for the goods listed under the export agreement.

Then the key concepts of the business sector, what we refer to as the business domain, are identified and documented in a class diagram This diagram contains classes without operations and hence consists only of classes, attributes and relations. A part of the business domain of our transaction is represented on Fig. 2.

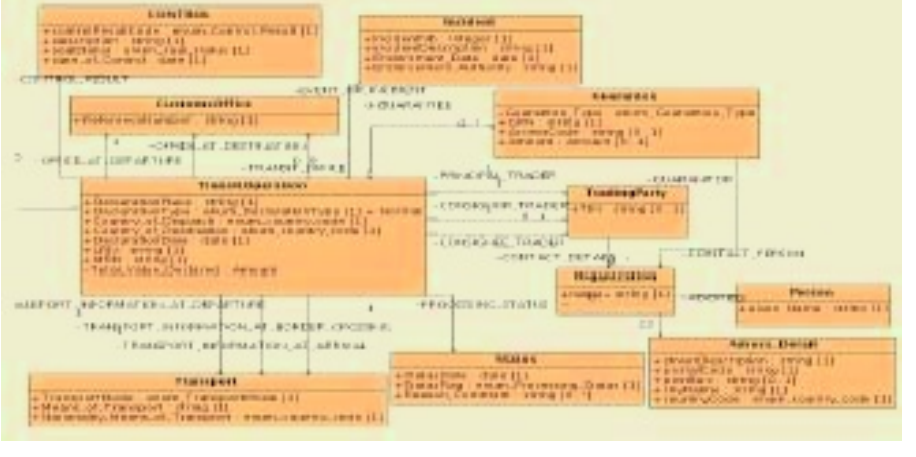

Fig. 2. Part of the NCTS Business Domain

3.2 Dynamics of the Transaction: Activity Diagram

Once the basic blocks have been described, the transaction is further specified with more details: the model of a transaction is composed of one activity diagram that describes the flow of business messages among its participants, accompanied by one class diagram for each message exchanged. The activity diagram is composed of:

- Swim-lanes representing the different roles of the transaction
- Activities performed by these roles
- Object flows representing messages exchanged. Each object flow must be preceded and followed by activities belonging to different swim-lanes, these swim-lanes being associated with the role sending and the role receiving the message respectively
- One initial state
- Only one final state (to avoid misunderstandings with workflow formalisms)
- Pseudo-states like forks, joins, decisions and merges (for the same reason, shortcuts on these pseudo-states, like having more than one transitions arriving to or leaving an activity, are forbidden). Only "reported" decisions are supported at the moment: they are decisions without any guard condition, allowing users to choose between 2 or more alternative messages.

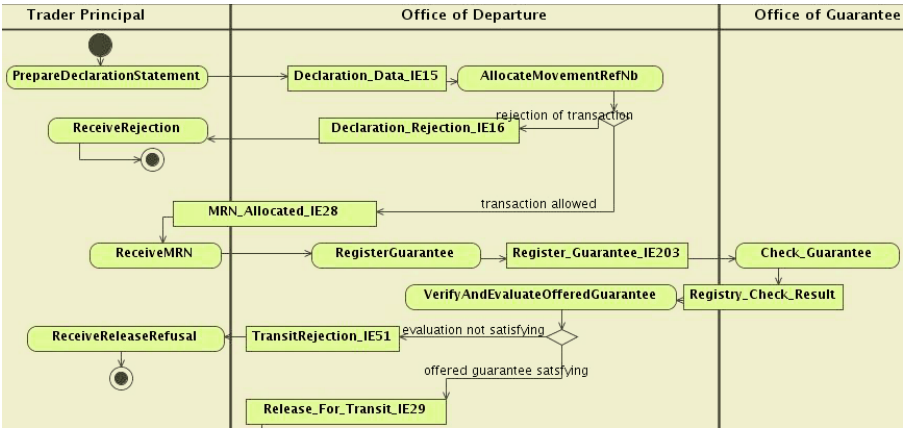

Fig. 3. Beginning of the NCTS transaction

Fig. 3 represents the first part of the NCTS activity diagram: the principal trader submits an export declaration to the Office at Departure, which allows or rejects the transaction. If permission is granted, the Office of Guarantee issues a guarantee and the transaction may go on.

3.3 Structure of Messages: Class Diagrams

For each message of the transaction, we define what information it must include depending on the information needs of the recipient at this stage of the transaction. To

do so, we have chosen to use a limited version of UML class diagrams, containing only constructs that can be used to represent XML messages. In order for an XML implementation to be possible, a class diagram must be limited to include only hierarchical relationships, that is, no loops and a unique root class that specifies the root of the message.

Class diagrams we consider contain:

- A "root" class, meaning a class with no navigable relation incoming.
- Classes with attributes of type UML data-types or of another class
- Relations between classes: only binary oriented associations, compositions or aggregations, and generalizations. These relations can have role names, and a multiplicity.
- Possibly additional constructs, like the "enumeration" stereotype, or "XOR" constraints

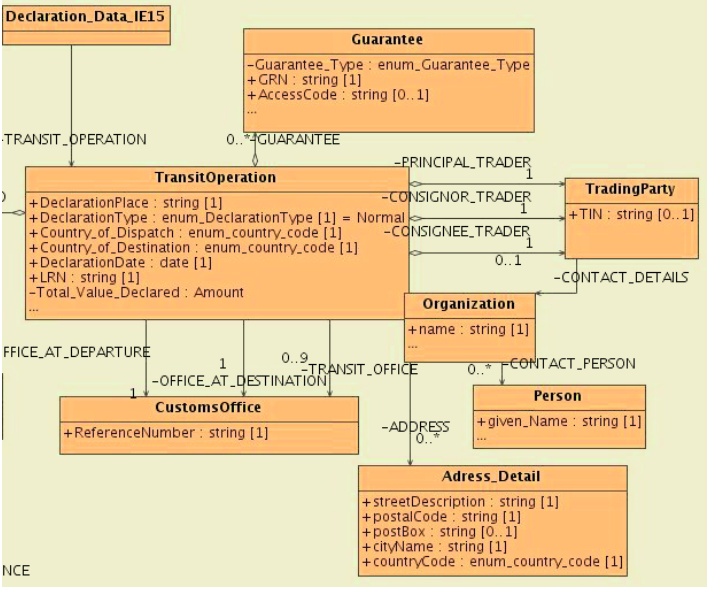

Fig. 4. Part of the class diagram of the "Declaration_Data_IE15" message

The class diagrams associated with the different messages are directly built from the global business domain diagram showed in Fig. 2. The information included in the message is a subset of the business domain information.

A part of he class diagram representing the message called "Declaration_ Data_IE15" of the NCTS transaction is shown on Fig. 4. This message contains information about the transit operation, the type of guarantee, the trading parties, and the cargo to be exported.

4 Expressing Business Rules Associated with a Transaction

The Oracle DataBase community has taken over the term "business rule" to designate any data rule than could not be directly represented in their relational database. This table-based rule representation sometimes leaded to a splitting of one rule in many places of the system. We situate ourselves more in the context of the "logical business rules" community founded by Ron Ross and Terry Moriarty using the name "The Business Rules Group" [15]. Their perspective on business rules is quite similar to what a traditional systems person might call a "system requirement": *trying to formalize what a system should do* (in English from their point of view).

A business rule is part of an accurate description of a transaction. Such a rule states upon specific aspects of this process to lead business participants in doing the right choices.

Formally speaking OCL [12] is the language recommended for expressing rules besides UML diagrams (see e.g. [2]). In practice we have experienced two problems with OCL. On the one hand, the language is not readable at all by business experts, on the other hand the validation of XML documents with regard to such specification is a useless technical challenge. As a conclusion, we have tried to identify the required subset of these rules and choose an ad-hoc formalism for expressing them.

We decided to differentiate 2 kinds of rules: simple, repetitive rules that link the values of elements between different messages, further called inter-message rules, and full-featured business rules allowing the expression of more complex rules.

4.1 Inter-message Rules

Inter-message rules are a means to link classes, association ends or attributes of different messages together. They are associated with an element of a class diagram describing a message and specify how the value of this element is related to the content of an element that occurs in a previous message of the current transaction. This kind of rule is especially useful in transactions where most of the data is passed from one message to its successors. Inter-message rules are specified differently from business rules, using simple UML notes, because that makes them much easier and quicker to be added, as we expect that message diagrams contain many such rules.

Some default inter-message rules can be generated automatically by the Efficient plug-in, based on an algorithm that creates a rule each time an element from a previous diagram is re-used with the same attributes and outgoing relations. These default rules can then be modified by the business expert according to his/her needs.

We have differentiated two kinds of rules: the “references” rules express that the value of the referenced element should be recopied, while the “cardinality reference” rules mean that the cardinality of the referenced relation should be preserved. We also have defined optional variants of these rules, which only serve as a help to pre-fill messages but are not enforced. These rules are written as notes on the UML class diagrams, stating the kind of the rule that applies to the linked element, and the “path” (in XML style) of the referenced element on the other diagram. We also have built a small user interface to choose these elements by choosing from a list of classes, associations and attributes.

As you can see on Fig. 5, there are two “reference” rules described by notes containing the name of the rule and the “path” of the referenced element in the “Declaration_Data_IE15” message. Here, it means that the Local Reference Number (LRN) of the operation must be the same in the two messages, and also that the Office at departure must be the same. This message also contains other rules not shown on this figure.

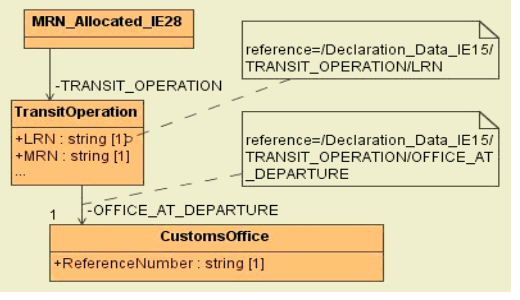

Fig. 5. Inter-message rules on the "MRN_Allocated_IE28" message

4.2 Complex Business Rules

At the level of the expression of more complex rules, the challenge of choosing a good formalism is twofold: on the one hand the rule needs to be *understandable by business experts*, to help them communicating, on the other hand it should be easily parsed and *understood by computers*. We therefore propose to interpret the same rule in a natural language form (English and French are already supported), and in a computable form (namely Xlinkit, an XML representation of first order logic predicates). The grammatically correct natural language interpretation of the rule is achieved thanks to the open source *Grammatical Framework* (GF) tool of Chalmers University of Technology in Göteborg [7], a strong-typed generic grammar framework.

The "business rules" to be used in the Efficient framework is composed of a *condition* and the (right) *behavior* that responses to such a condition occurrence. The condition describes a situation of the "real life" that may occur during the transaction. We propose to include any factor of the environment that may trigger the transaction as a very part of it, that is: as information provided by some actor of the transaction.

The reaction to a condition occurrence is specified as a particular flow of the transaction, in the UML activity diagram described in Fig. 3, while the way of expressing and computing conditions is detailed below.

Business Rules Facts and Conditions

For the sake of ease, a rule is being built in a progressive refinement of a tree-like structure, directly mapped onto its natural language meaning to be sure the refinements chosen by the business experts match his intentions. This is done using the graphical user interface of GF [7].

A business rule ranges over *business facts* (see [15]) that represent constructs of the real world in terms of information available during the transaction. Many business facts are derived from the specification of each message, and hence they differ from a transaction to the other. Facts include, among others:

- the simple occurrence of a document
- the exact time at which this document was sent
- any field (element) of the sent document
- any field of the document that is of same kind

From the document of Fig. 4 we propose, for instance, "the name of any Organization" as a fact that describes the name of any of the mentioned traders. The above business facts are combined using typed operators, either common in the business all-day-life as:

- a document or field existence
- a maximal or minimal delay between documents or dates
- a date expiry

or computed using classical arithmetic, boolean and set operators. The case all operators are typed prevents the business expert to refine rules that would not be computable, while the simultaneous natural language interpretation of the rule ensures its semantics.

Business Rule Context
Technically speaking, a rule may apply under some circumstances (only during a working day, for instance). Those circumstances form what we call the *context* of a rule, users of the Efficient framework may use the facts detailed above to specify a particular context, or simply state that the rule applies either

- always
- after or before a particular document is sent
- between the occurrence of 2 documents (when a rule is triggered *and* expires)

User Interface
The graphical interface provided by GF displays the current state of the tree-form of the rule, as well as its natural-language interpretation, while giving the ability to further refine the rule by selecting the appropriate element. Here is an example rule, shown in its natural-language form by the tool: "The 'StatusDate is not expired' rule constraints the current transaction after a Declaration_Data_IE15 document is sent. *StatusDate of Status in Declaration_Rejection_IE16 is not expired*."

We easily identify the rule context at the beginning of this statement.

4.3 Generation of Code for the Business Rules

As explained in Section 2, all the code associated with a transaction is fully and automatically generated from the specification models. Details about the generation of code associated with XML Schemas, workflow procedures and graphical GUI can be found in [5]. It follows according the following principles, that are fully compatible with the ebXML initiative:

- for the structure of messages, we chose to use W3C XML Schema files [24], because XML Schema is a well-known format from the W3C and because there are many open source tools available for data validation and manipulation.
- For the flow of messages, we use a format that is understood by the workflow engine, and that is based on a standard: XPDL (XML Process Definition Language, [25]), from the Workflow Management Coalition [19].

Regarding the code associated with complex business rules, the tree structure that represents a business rule may be interpreted either in English, French, or any other interpretation for which a mapping exists to Xlinkit.

Xlinkit [22] is used to verify consistency constraints that have been defined for a set of XML documents, in our case the XML messages exchanged. Xlinkit implements the most basic logical predicates of the set theory, relying on XPath expressions for the definition of sets of nodes in documents.

First Order Logic Mapping

The interpretation of a business condition as a first order logic (FOL) predicate is quite straightforward, as illustrated below on the same rule as before.

- a business fact is mapped onto its corresponding XPath expression.
- a condition that may trigger the rule (defined in its context) is translated as a premise of a logical implication operator, whose conclusion is the below operator combination.
- a typed operator is mapped either onto its FOL equivalent if there is one, or onto a function call that implements it.

Xlinkit provides extension mechanisms that allowed us implementing some business-oriented operators (as a date expiration check) and integrating them as common function calls into the animation framework.

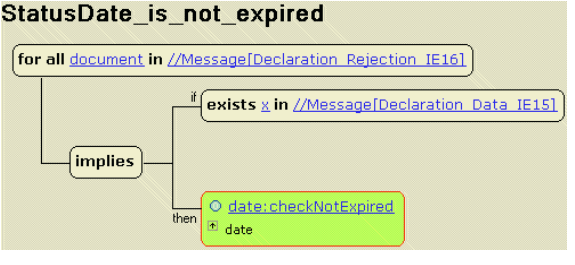

Fig. 6. 1st order logic interpretation of a business condition

5 Verification and Validation of Transactional Models

The development and the implementation of the Efficient formal modeling language pursues two objectives: the formal representation of the various aspects (flow, content, governance and rules) that impact on an e-business transaction and its

verification. In the context of the Efficient project, the validation of the e-business requirements is achieved by two complementary approaches:

- Automatic verification: the approach based on a proof system aims at verifying that a given property is verified for each possible scenario of the system. . A scenario is associated with one possible execution of the transaction, i.e. one possible path in the activity diagram. In particular, it can be used to verify that the restrictions on the UML language imposed by the Efficient framework are respected in the model, and that there are no logic errors in the model.
- Validation through animation: the animation of transactions aims at discovering errors before the actual implementation of the transaction. Such errors include deadlocks, live locks, missing or non-adequate information content, missing messages or wrong flow of messages, etc. It is the business experts themselves who check the validity of the transaction by visualizing the execution of a set of scenarios that involves the different messages. They simply play the roles of the transaction's actors by receiving messages and sending answers.

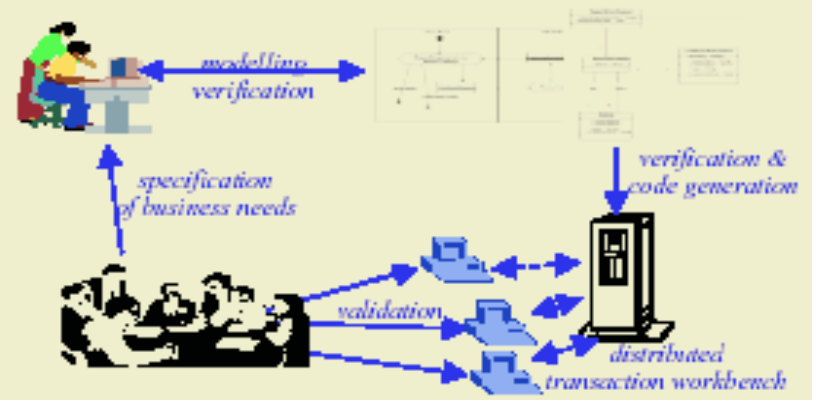

Fig. 7. The Efficient verification and validation activities

5.1 Formal Verification

Although the animation approach is well suitable for business experts, its efficiency depends on the relevance of the scenarios chosen. The effectiveness of this approach decreases with the complexity of models that are to be verified: it becomes difficult to find suitable test cases for discovering errors. Besides, this approach can only detect the presence of failures, but cannot prove the correctness of the system we are testing. Contrary to this approach, automatic verification can discover errors that may not be found in the first step because of defects of the test cases. Nevertheless, it is not yet suitable for business experts and thus cannot be used to verify that the model corresponds to their needs.

In the two following subsections, we report about some properties that define a correct e-business process, and then we illustrate their verification approach.

Properties of Correct e-Business Processes

After studying different kinds of errors discovered during validation by animation, three property categories have been elected:

1. Structural properties: this type is closely related to the topology of the diagrams. Structural properties can be directly verified on the UML diagrams without animating the transactions. They are necessary conditions that ensure the feasibility of the transaction. If one of them is not fulfilled, we can assert firmly that the transaction must be erroneous. The structural properties that must be verified by an e-business transaction can be classified into two categories. The first category facilitates the definition of the formal semantics for the class and activity diagrams that make up the transaction. This category comprises the following properties:

   a) A class diagram must not contain circuits. The algorithm used for generating the XPDL code imposes this property.
   b) Each activity diagram must have one initial state and one to several final states. The first part of this property determines the starting point of the transaction; the second specifies its end point. Reaching a final point means that the transaction has been fully accomplished.
   c) The number of incoming/outgoing transitions of each activity is equal to one.

   The second category corresponds to general reachability properties. This category includes the following properties:

   a) Each node must be reachable from the initial node. This property ensures that there are no superfluous nodes on a diagram.
   b) From each state, there must exist a path such that one of the final states can be reached. This property ensures that whatever the node reached during the execution of transaction, it may be possible to get to a final state: the system will hence correctly terminate.
   c) To avoid circular dependencies, we assume that for each input node *A* of a join node *C*, there must not exist a path from *C* to *A*. In fact, on the one hand, the fact that *A* is an input node of *C* implies that the node *C* will follow the node *A*, on the other hand, a path form *C* to *A* would imply that the node *A* will follow node *C*: detection of a deadlock.

2. Real time properties: these properties aim at evaluating the system in terms of their time responsiveness. Placing constraints on the different activities, we want to be able to determine if the system can answer in a limited amount of time. The different type of time constraints that can be stated on activities include delay, duration, timed events of the form When (t) or After (t), etc.

3. Dynamic properties: the dynamic feature of these properties means that they are related to two or more states of the system. Its verification is achieved on a set of

states describing a possible evolution of the system. Among the possible set of dynamic properties, we cite the two following safety /liveness properties:

a) Each guard associated to a transition is not always false (a contradiction). This property is used to eliminate dead paths.
b) In most domains, applications are desired to be deterministic, that is, at each moment; at most one behavior is possible. For that reason, we impose that the guards must be disjoint.
c) The disjunction of all the guards yields TRUE. This property is used to eliminate a potential deadlock.

Our Verification Approach

The verification process begins by checking the structural properties according to the approach we have defined in [8]. This approach is based on the USE approach fully described in [13]. If these properties are fulfilled, then the real time properties are checked by following a formal approach based on the PROMELA language and its tool support SPIN [10]. More details about this approach can be found in [9]. The reason for checking the structural properties at the beginning of the verification process lies in the fact that these properties are rather simple and fast to verify. Moreover, they allow us to rule out a wide range of ill-defined transactions without getting involved in a deep semantics analysis that may be very time consuming. Fig. 8 depicts the different verification phases.

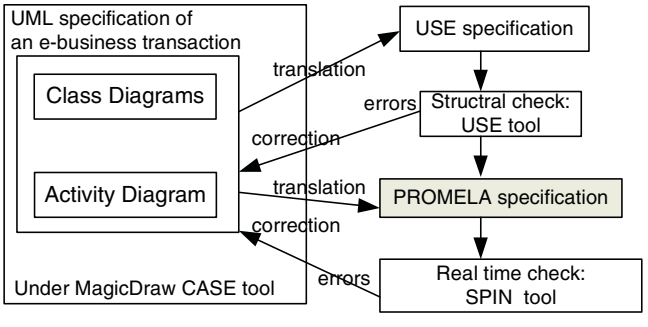

Fig. 8. The verification process

At current time, the verification of structural properties, which is integrated as a Plug-in in the MagicDraw CASE tool, is fully automated. The tool states the number of properties that were verified, including those that were not satisfied, along with the time consumed by the verification task.

5.2 Validation Through an Animation

The animator tool orchestrates a process in a distributed scenario: it allows the business users to exchange messages according to the activity diagram of the transaction. When a message is sent, its structure is checked, along with the validity of the business rules specified for the message.

The animator tool is composed of a set of modules, which are integrated around a workflow engine that orchestrates the transaction. The process to be executed by the workflow engine is based on the process modeled in the activity diagram, enriched with the following additional activities (see Fig. 9):

- Before a message can be sent by a role, an additional activity is introduced informing the different participants about the messages that they can send.
- The reception activity is called when the message is received. It stores the XML message received in a XML database
- Then the structure of the message is checked as well as the associated business rules, in a "check" sub-process (see Checker sub-flow on Fig. 9).
- If the checking returns an error, an error message is sent to its sender
- Otherwise, the message is forwarded to its recipients (with eventually pre-filled values, depending on the inter-message rules associated with it)

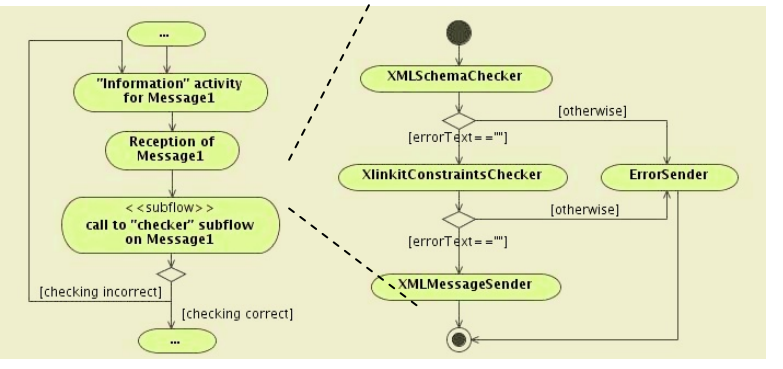

Fig. 9. Process for each message in the animator, and checker sub-flow

Efficient provides the recipients with a web based client tool that allows them to:

- Instantiate one of the processes for which the animator is configured, by identifying potential participants along with the roles they play in the transaction
- Receive messages addressed to one of the roles they play (using web services)
- Send messages, after the animator has executed a corresponding "information" activity, by using web-based forms to input data and web services to send it. The forms are XForms files [21], and are displayed using an open source XForms implementation called Chiba [1]
- See the list of previously received messages

In general, in the animator tool and in the client, we have favored XML standards over other formats, as well as the use of open source tools over proprietary tools. For example, the animator uses an open source workflow engine called WFMOpen [20], and use different XML libraries to parse and validate XML files.

6 Conclusion

This article introduces the Efficient toolset, a comprehensive environment for designing and evaluating e-business transaction. Efficient facilitates the communication between business experts and IT analysts and allows them to progressively elaborate the specification by generating a prototype of what the transaction might look like, without having to implement it. Current and future work will focus on the enhancement of the expressiveness of the modeling language: choices between two or more activities may be e.g. based on the actual values of a message form, a more flexible definition of roles will be investigated on, and we are going to evaluate the use of time constraints. Also, an upgrade to UML 2.0 will be considered. From the verification point of view, we plan to extend the developed verifier tool to take real-time features into account. Another interesting direction is to consider nested transactions. The concept of "nested transactions" allowing a specification reuse, it would be interesting to study how the correctness of a nested transaction can be established (or deduced) from the correctness of the sub-transactions that compose it.

References

1. Chiba, open source Xforms implementation, http://chiba.sourceforge.net
2. Corréa et C. Werner, *Precise Specification and Validation of Transactional Business Software*, in Proc. of IEEE Joint International Requirements Engineering Conference (RE'05), Kyoto, Japan, 2005.
3. ebXML, http://www.ebxml.org/
4. R. Eshuis, P. Brimont, E. Dubois, B. Grégoire, S. Ramel, *EFFICIENT: A Tool Set for Supporting the Modelling and Validation of ebXML Transactions*, poster and short paper at ESEC/FSE 2003, http://efficient.citi.tudor.lu/
5. R. Eshuis, P. Brimont, E. Dubois, B. Grégoire, S. Ramel. *Animating ebXML Transactions with a Workflow Engine*, In Proc. CoopIS 2003, volume 2888 of LNCS, Springer, 2003.
6. extensible Markup Language, http://www.w3.org/TR/2004/REC-xml-20040204/
7. *Grammatical Framework*, open source tool, http://www.cs.chalmers.se/~aarne/GF/
8. N. Guelfi, A. Mammar, B. Ries, *A Formal Approach for the Specification and the Verification of UML Structural Properties: Application to E-Business Domain*, International Workshop on Software Verification and Validation (SVV 2004), workshop of ICFEM'04, Seattle, WA, USA, 2004.
9. N. Guelfi, A. Mammar. *A Formal Approach for the Verification of E-Business Processes Using the PROMELA Language*. Submitted to FASE'05.
10. G.J. Holzmann, *The Model Checker SPIN*, Software Engineering 23(5)pp.279-95, 1997.
11. MagicDraw, commercial tool, http://www.magicdraw.com/
12. Object Constraint Language (OCL), http://www.omg.org/cgi-bin/doc?formal/03-03-13
13. M. Richters, *A Precise Approach to Validating UML Models and OCL Constraints*, PhD Thesis, University of Bremen, 2001.
14. RosettaNet Partner Interface Program (PIP), http://www.rosettanet.org
15. R. Ross, *The Business Rule Book: Classifying, Defining and Modeling Rules*, Second Edition, 1997.

16. M. Schmitt, B. Grégoire, C. Incoul, S. Ramel, P. Brimont and E. Dubois, *If business models could speak! Efficient: a framework for appraisal, design and simulation of electronic business transactions*, ICEIMT 04, http://efficient.citi.tudor.lu
17. UN/CEFACT Modeling Methodology (UMM), http://www.ebxml.eu.org/umm.htm
18. Unified Modeling Language, Object Management Group specification, http://www.omg.org/uml
19. WfMC, Workflow Management Coalition, http://www.wfmc.org/
20. WFMOpen, open source workflow engine, http://wfmopen.sourceforge.net/
21. Xforms, W3C standard, http://www.w3.org/TR/2003/REC-xforms-20031014/
22. X*linkit*, commercial tool, http://www.systemwire.com/xlinkit/
23. XML Common Business Library (xCBL), http://xml.coverpages.org/cbl.html
24. XMLSchema, W3C standard, http://www.w3.org/XML/Schema
25. XPDL, standard from the WfMC http://www.wfmc.org/standards/XPDL.htm,
26. W.M.P. van der Aalst, H.M.W. Verbeek, and A. Kumar, *XRL/Woflan: Verification and Extensibility of an XML/Petri-net based language for interorganizational workflows,* http://tmitwww.tm.tue.nl/staff/wvdaalst/publications/p139.pdf
27. Nick Szirbik, Gerd Wagner, *Steps Towards Formal Verification of Agent-based E-Business Applications,* http://www.informatik.uni-hamburg.de/TGI/events/ moca01/ wagner-final.pdf