Steps Towards Formal Verification of Agent-based E-Business Applications

Nick Szirbik, Gerd Wagner
Dept. of Information & Technology, Eindhoven University of Technology
Postbus 513, 5600 MB Eindhoven
{g.wagner; n.b.szirbik}@tm.tue.nl

Abstract

In this preliminary report about ongoing work, we discuss the issues of formalizing the specification of interaction process types for e-business applications and identifying techniques for *proving correctness properties* for them. We introduce the concept of *coherence* for interaction processes in the broader context of multiagent communication based on agent communication languages. Interaction process types can be sepcified by means of reaction rules in a declarative way. In order to make use of the formal analysis techniques established for Petri nets, we investigate possibilities how to translate reaction rules into Petri nets.

1. Introduction

In a world characterized by rapid change, e-business applications have inherently a very short life-cycle, short time-to-market, must be adaptable for upgrade and have to follow the changes in the business processes that are supported by these application. Their development would be less error-prone if it would be possible to use formal verification techniques implemented with automated reaoning technologies. Formal verification is well suited for systems with a relatively small state space, or where the essential aspects of the system can be captured in a model that may be formally checked. To validate the final application, testing and other validation techniques are used. However, the validation phase can be extremely costly, and formal verification can help to avoid major errors in the design, reducing the costs of the validation phase. Also, verification can "catch" problems which are extremely difficult to be identified in the validation phase and usually appear late after the deployment of the application, incurring losses for the user and costly maintenance for the application provider.

We discuss in this paper the issue of *coherence* in interaction processes, which subsumes a certain set of properties that we claim to be important to be checked for e-business applications. The paper is structured around this concept; the next section explains relevant aspects of the agent paradigm, section 3 is presenting the coherence concept, section 4 is explaining the action/event and claim/commitment concepts and their usage in supporting coherence, section 5 is about the rule specification used in the paper, section 6 explains how some particular types of rules can be translated into Petri Nets, section 7 is presenting how inter-agent communication can be modelled, section 8 puts some of the questions that have been raised by this reasearch and section 9 concludes the paper. Some ideas above potential future work are also given there.

2. Agent-Oriented E-Business Modeling and the Issue of Verification

The objective of the research presented in this paper is to analyse the problem of, and develop a method for, how to verifiy the coherence of e-business processes as automated interaction processes across different orgaizations. For obtaining high-level concepts of interaction and communication, an agent-oriented approach to e-business modeling seems to be most promising. *Agent-Orientation* is emerging as a new paradigm in software and information systems engineering. It offers a range of high-level abstractions that facilitate the conceptual and technical integration of communication and interaction with established information system technology. Agent-Orientation is highly significant for business systems since business processes are driven by and directed towards agents (or actors - we treat both terms as synonyms), and hence have to comply with the physical and social dynamics of interacting individuals and organizations. While today's enterprise information system technology is largely based on the metaphors of data management and data flow, Agent-Orientation emphasizes the fundamental role of actors/agents and of communication and interaction for analysing and designing organizations and organizational information systems. This turns out to be crucial for a proper understanding of business processes and their underlying rules.

Interaction processes are determined by the behaviour of the participating agents. The behaviour of natural (and artificial) agents can be described (and specified) by means of rules. Business rules are the policies and constraints of the business, whether the "business" is banking, software development or health care. The Object Management Group simply defines them as "declarations of policy or conditions that must be satisfied". They can be expressed in ordinary language (possibly using special templates) or by means of a formal language. An interaction process type can be specified by defining, for each participating party, a set of inter-related rules. The formal language for such rules must include facilities for incorporating agent communication languages, message content languages, and domain ontologies in a flexible (suitably parameterized) manner.

There are two main questions to be answered:

1) what properties are worth and essential to be verified in such a system
2) from what details can we abstract away in order to make verification feasible (i.e. computationally tractable).

The verification method can employ different techniques.

E-business applications are supporting interaction processes between various types of enterprises (viewed as *institutional agents*). An interaction process can be defined in many ways, but we prefer here a definition based on the agent paradigm. We interpret this paradigm in a way that emphasizes that all the participating active entities are agents. The agents can be generally classified in five classes: human, institutional, software, robots and animals, but only the first three are interesting to our discussion here. E-business processes can be semi-automated or fully automated.

It is well-known that the process formalism of Petri Nets can be used for verifying certain properties of business processes [Bas98, Aal00, Aal01]. We discuss the possibility to translate a rule-based process specification into a Petri net An earlier attempt to translate rules into Petri Nets has been made in [Naz93].

3. Coherence in Interaction Processes

An interaction process is a temporally ordered set of *events*, *actions* and *activities* perceived and performed by agents, following a *protocol* that specifies the *type* of the interaction process. An interaction protocol can be specified by a set of *reaction rules* that determine the behavior of the involved agents. Business interactions, for example, comprise communications (message exchange), payment transactions, product deliveries, and the performance of services. An interaction process

may be fully automated, e.g., in electronic commerce supply chains, in agent-mediated group scheduling, in robotic soccer games (RoboCup), in Automatically Guided Vehicle (AGV) Systems, and in other types of artificial multiagent systems. In many cases, however, multiagent systems consist of institutional, human and artificial agents, and interaction processes in such mixed systems are only semi-automated, as in an enterprise where employees ('human agents') interact with organizational units ('institutional agents'), with agentified information systems ('software agents') and with agentified machines ('robots'). In any case, agents have to interact with each other in a flexible but coherent manner.

Examples of incoherent interactions in multiagent systems are:

- mis-understanding the message of a team mate in a robotic soccer game, or not giving way to another AGV although it has communicated a higher order priority;
- not making any compensation offer when being unable to deliver an item after having acknowledged the purchase order for that item, or sending a payment reminder although the payment has already been made, in an electronic commerce supply chain;
- or cancelling an appointment with a higher priority in favour of an appointment with a lower priority in agent-mediated group scheduling.

Incoherence may be caused by plain human user errors, by programming errors, by miscommunication, and by unforeseen changes in the environment of an agent. While coherence is a real concern in all types of interaction processes, it seems of particular importance for inter-organizational processes that are automated or semi-automated by means of information technologies.

The coherence problem in multiagent systems covers a larger domain than the sub-problem we try to solve in the first step of our research. On a higher level the coherence of (semi-) automated interaction processes consists of the following subproblems:

1. Enforcing *linguistic coherence*: avoiding miscommunication by speaking the same language and sharing a common understanding of its terms. This requires to use
   - a standard communication language (such as FIPA-ACL or ebXML) that defines message types and their communication semantics,
   - standard composition languages for expressing composite propositions and actions, and
   - shared ontologies (corresponding to domain models) that define the basic vocabularies used for denoting entities and their properties and relationships.
2. Enforcing the *normative coherence* of interaction processes as regulated by commitments, claims, and norms (where norms can be conceptualized as meta-commitments, i.e., as commitments referring to other commitments, see [Sin99]). The operational semantics of commitments includes procedures that define how they are created and discharged, under which (exceptional) circumstances they can be cancelled, and what happens if they are violated. Norms and commitments constrain interaction processes. Unlike integrity constraints, however, they may be cancelled and may be violated without crashing the system.
3. Enforcing *process integrity* by
   - verifying the correctness of interaction protocols (based on safety and progress properties) at design time, and by
   - checking interaction constraints and handling exceptions at runtime.

Current and forthcoming XML-based standards for electronic business communication, such as the Electronic Business XML (ebXML) initiative by UN/CEFACT and OASIS, do not address the

coherence problem in general, but are focused on limited forms of linguistic coherence. Only ebXML defines that a 'Collaboration Protocol Agreement' must exist between two parties in order for them to engage in automated interactions. Since the ebXML Business Process Specification Schema, currently still under development, is based on the rather limited and implementation-oriented definition of events and actions in UML, one may expect that the forthcoming ebXML concept of interaction protocols will not be able to capture the semantics of (semi-)automated interactions, including a proper treatment of the issues of normative coherence and process integrity.

In order to attack the problem of process integrity for fully automated interaction processes (where no human agents are involved), we would like to investigate the possibilities to translate interaction rules into Petri nets and use the well-established process verification techniques of Petri nets. The problem to be solved is how to translate a rule set into a Petri Net that preserves its operational semantics.

4. Actions and Events, Commitments and Claims

In a business domain, there are various types of actions performed by agents, and there are various types of global state changes, including the progression of time (which is perceived usually as a discrete value). For an external observer, both actions of business agents and environmental state changes constitute events. In the internal perspective of an agent that acts in the business domain, the actions of the other agents count as events but not the actions of the agent itself. In the external perspective, actions create events, but not all events are created by actions. Those events that are created by actions, such as delivering a product to the customer, are called *action events*. Examples of business events that are not created by actions are: the fall of a particular stock value below a certain threshold, the sinking of a ship in a storm, etc. In our approach, we take into consideration only the action events.

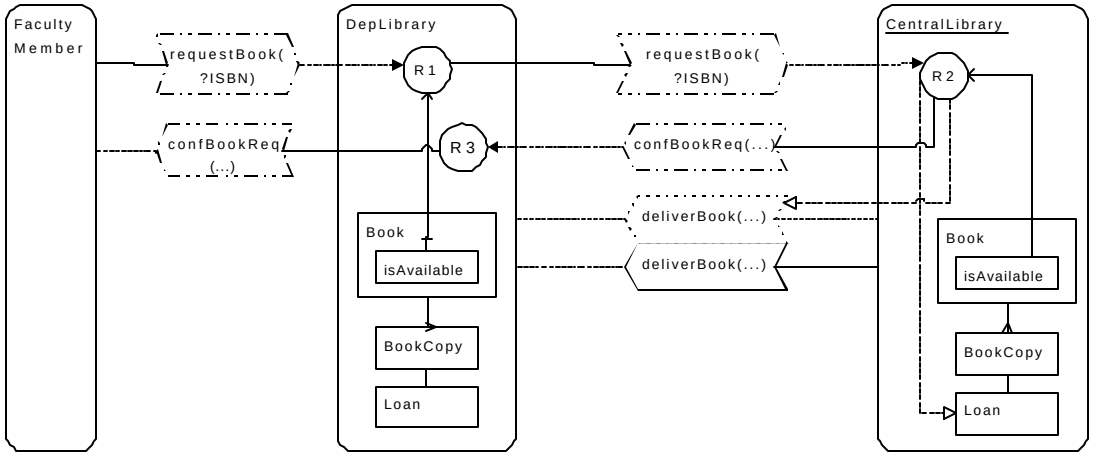

Figure 1: An interaction pattern diagram describing the process type where faculty members request books from a department library, and where the request is forwarded to the central library, since the books are not available.

We also make a distinction between *communicative* and *non-communicative* (e.g. physical) actions and events. Many typical business events, such as receiving a purchase order or a sales quotation, are communication events. Business communication may be viewed as asynchronous point-to-point message passing. The expressions receiving a message and sending a message may be

considered to be synonyms of perceiving a communication event and performing a communication act. This is different from the low level concept of method invocation in object oriented programming, because agent-oriented programming (AOP) assumes the high level semantics of speech-act-based Agent Communication Languages (ACL) messages [Wag00a].

The second fundamental concept in business interaction processes is the *commitment* concept. Representing and processing commitments and claims in information systems explicitly helps to achieve coherent behaviour in real-life interaction processes. In [Sin99], the social dimension of coherent behaviour is emphasised, and commitments are treated as ternary relationships between two agents and a "context group" they belong to. For simplicity, we treat commitments as binary relationships between two agents. Commitments to perform certain actions (or to check if certain conditions hold) arise from specific communication acts. For instance, sending a sales quotation to a customer commits the vendor to reserve adequate stock of the quoted items (for some time). Likewise, acknowledging a sales order implies the creation of a commitment to deliver the ordered items on or before the specified delivery time.

These two concepts are depending on the perspective chosen. In the perspective of a particular agent, *actions* of other agents are viewed as *events*, and *commitments* of other agents may be viewed as *claims* against them. In the internal perspective of an agent, a commitment refers to a specific action to be performed when necessary for another agent (due to time for example), while a claim refers to a specific event that must be created by an action of another agent, and has to occur in the future. In the view of an external observer, actions are also events, and commitments are also claims, exactly like two sides of the same coin. This means that in any model, a *commitment* of agent a1 towards agent a2 to perform an action r, can also be viewed as a *claim* of a2 against a1 that an action r will be performed. There is no similar coupling between actions and events, because an agent can sense an event, but due to its internal status, it may, or may not take any consequent action. It is simpler to model claims and commitments in a Petri Net, due to their mirrored nature (these are created and dismissed in pairs) and the related semantic is usually very clear and simple to represent.

5. Reaction Rules

In the *Agent-Object-Relationship* (AOR) modeling language proposed in [Wag00a,Wag00b], reaction rules may be visualized in *Interaction Pattern Diagrams* including action events and claim/commitments. An example of such a diagram is shown in Figure 1 where reaction rules are visualized as a circle with incoming and outgoing arrows drawn within the agent rectangle whose reaction pattern it represents. Each reaction rule has exactly one incoming arrow with a solid arrowhead: it represents the triggering event condition which is also responsible for instantiating the reaction rule (binding its variables to certain values). In addition, there may be ordinary incoming arrows representing state conditions (referring to corresponding instances of other entity types). There are two kinds of outgoing arrows. An outgoing arrow with a dashed line denotes a mental effect (i.e. a change of beliefs and/or commitments). An outgoing connector to an action event type denotes the performance of an action of that type. For instance, the reaction rule R2 in Figure 1 reads as follows: *When the central library receives a certain book request from a department library, it checks if a copy of that book is available, and if this is the case, the request is confirmed, and a new corresponding loan object as well as a commitment to deliver the requested book in due time is created.*

A reaction rule consists of an antecedent expression (its 'body') and a consequent expression (its 'head'). The antecedent is composed of two parts: an event condition (where an event is represented

by an incoming message), and a state condition (to be checked against the state of the agent that is executing the rule). Also, the consequent is composed of two parts: an action term and an effect formula that specifies the state changes effected by the action performance. We use a tabular notation for specifying reaction rules, such as in Table 1.

ON	**Event**	*book request from a department library*
		`RECEIVE` requestBook(?ISBN) `FROM` ?DepLib
IF	**Condition**	*a copy of that book is available* BookCopy.isAvailable(?ISBN, ?InvNo)
THEN		
	Action	*confirm the request*
		`SEND` confBookReq(?ISBN)
	Effect	*a new corresponding loan object as well as a commitment to deliver the requested book in due time is created*
		`CREATE COMMITMENT TOWARDS` ?DepLib `TO` deliverBook(?ISBN) `BY` tomorrow(); `CREATE BELIEF` Loan(?DepLib, ?ISBN, ?InvNo, today()

Table 1: The interaction rule R2 of Figure 1 in tabular form.

6. Translating Reaction Rules into Petri Net Transitions

Since reaction rules specify state transitions, it seems natural to relate them to Petri net transitions. The state of an e-business scenario consists of the private state components of each agent (such as its privately held beliefs) and of the public state components which are shared among two or more agents, such as the current action events and commitments/claims. We are interested in those cases where the state of a scenario can be represented by means of a certain token population of the places of a (preferably classical) Petri net, and the behavior (i.e. the reaction rules) of the involved agents can be represented by means of transitions in this net.

In order to obtain classical Petri nets, we have to make some severe restrictions:

- the state condition consists of a single *propositional variable* (or a conjunction of propositional variables)
- the event consists of an incoming *message without content parameters*
- the action consists of one or more outgoing messages without content parameters
- the effect consists of a single propositional variable (or a conjunction of propositional variables)

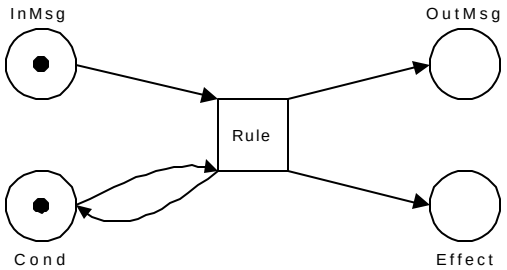

Figure 2

Under these assumptions we obtain the PN shown in Figure 2, where `InMsg` and `OutMsg` are places that represent message buffers, and `Cond` and `Effect` are places that represent propositions. Notice that, while a message is consumed and can therefore be represented as a token on an ordinary place, a state condition is normally not 'consumed' but rather preserved after firing the transition. The preservation of a condition can be achieved by a reproduction arc as in Figure 2, although this is not a very natural representation of what is really going on.

Since both a condition and an effect formula may be a negated proposition, we have to introduce *negative* input and output arcs as in Figure 3.

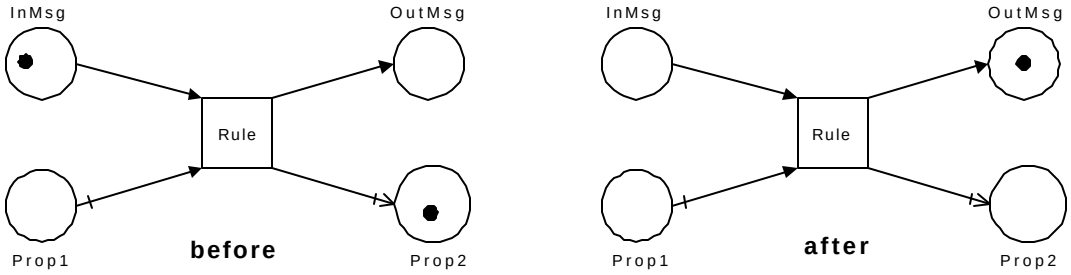

Figure 3: Firing a transition representing a reaction rule with a negative condition and a negative effect.

Thus, if the behavior of an agent is specified by a set of reaction rules satisfying the above restrictions, we can translate it into a Petri net whose transitions may be only linked through the propositional places (if we do not consider the technical possibility to send a message to oneself).

7. Multiagent Nets

For representing a complete process type involving several agent types, such as in Figure 1, we may translate the rule set of each involved agent type into a single agent Petri net such that these nets are linked to each other via the shared message buffer places. Notice that the single agent Petri nets must not be linked via private propositional places because they can be only accessed by their owner. We call the resulting type of Petri net a *multiagent net*.

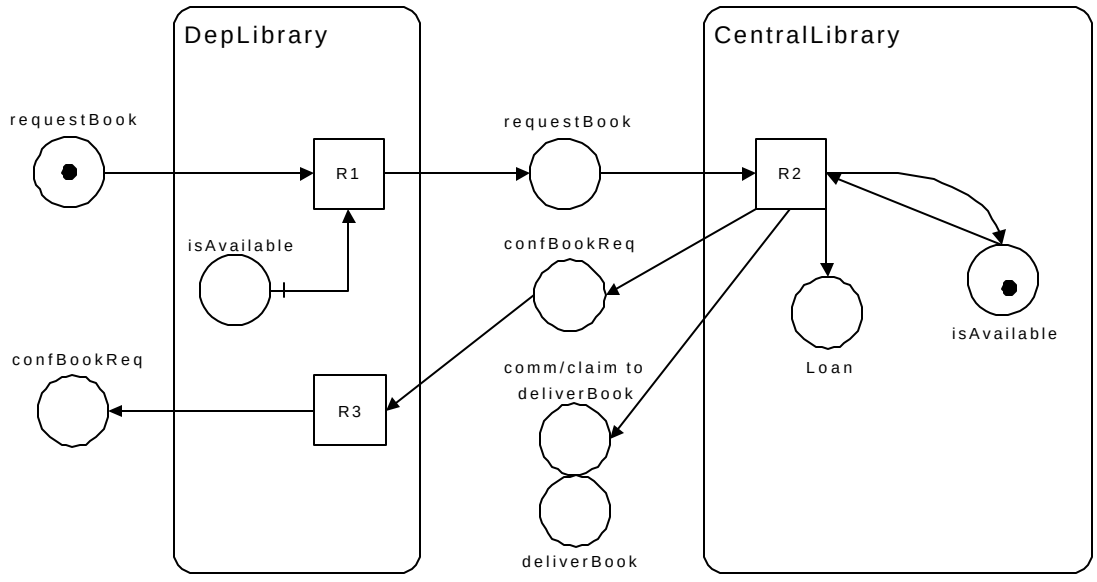

Figure 4: A multiagent net corresponding to the interaction pattern diagram of Figure 1.

The multiagent net shown in Figure 4 corresponds to the interaction pattern diagram of Figure 1. The two single agent nets are drawn in a rectangle with rounded corners. They contain the rules and the private beliefs of the respective agent. Both messages and commitments/claims are shared between two or more agents, so they are drawn outside of any agent rectangle.

Since we only have one type of place in the multiagent net, it is visually less clear than the interaction pattern diagram. However, we hope that by translating reaction rules into Petri nets, we will be able to benefit from the formal analysis techniques that have been established for them.

9. Conclusions and future work

This research is still in a preliminary stage. We presented in this paper some techniques how to translate an interaction process type specified by reaction rules into a Petri net. To perform experiment with agents we are using the PROVE system (see [Szi00] and [Aer00]) that bis based on the production rule engine JESS (Java Expert System Shell). The syntax of Jess rules is almost similar with CLIPS, and the rules described in the AOR diagrams must be translated (by a rule generator linked to the AOR enabled-editor) to obtain these in the format needed by the inference engine used by the agents. In this paper, for the sake of simplicity, we are using a tabular (schema-like) description of the rules. CLIPS code, based on predicate logic, is too cluttered to be easily readable.

It would be nice to include verification in a more general agent-oriented development method, where the system and the interaction processes are designed using the AOR modeling language, and the rules can be generated automatically in CLIPS format and also translated from the AOR description into Petri Nets. Probably, the second translation will be semi-automated, needing human intervention to define new predicates for linking the rules in the sequences dictated by the interaction process structure.

Another conclusion is that any multiagent system which can be described in the AOR modeling language, where the rules have a more generic structure, can be verified in this way. The problem is that the AOR specification of the system could take time, and it is probably simpler to translate the interaction process directly into Petri Nets. We still have to investigate which way is the most straightforward.

The next step in our research is to try to fully automate the translation process. Conversely, we want to know which are the best ways to specify the rules to make them easily translatable. One of the first guidelines is to try to program in a way which enables the chaining of rules. This is happening however in any rule-based system which is carrying out a process, but the condition/places are not explicitly specified.

References

[Aal00] Aalst, W.M.P.,vd, (2000), "Loosely Coupled Interorganisational Workflows: Modelling and Analysing Workflows Crossing Organisational Boundaries", Information and Management, vol. 32, no. 2, March 2000, pp.67-75.

[Aal01] Aalst, W.M.P., vd, (2001), "The P2P Approach to Interorganisational Workflows", Proc. of 13th Intl. Conference on Advanced Information Systems Engineering (CAiSE'01), Springer, Berlin.

[Aer00] Aerts, A.T.M., Szirbik, N.B., Hammer, D.K., Goossenaerts, J.B.M., Wortmann, J.C., (2000), "On the Design of a Mobile Agent Web for Supporting Virtual Enterprises", WETICE2000 Conference, Gaithersburg, NY.

[Bas98] Basten, T., (1998), In Terms of Nets: System Design with Petri Nets and Process Algebra, PhD thesis, Eindhoven University of Technology, December 1998, Eindhoven, The Netherlands.

[Naz93] Nazareth, D.L., (1993), "Investigating the applicability of Petri Nets for Rule-Based System Verification", IEEE Trans. on Knowledge and Data Engineering, vol5, no. 3, June 1993, pp.402-416.

[Sin99] Singh, M.P., (1999), "An Ontology for Commitments in Multi-Agent Systems", Artificial Intelligence and Law, no. 7, July 1999, pp.97-113.

[Szi00] Szirbik, N.B., Wortmann, J.C., Hammer, D.K., Goossenaerts, J.B.M, Aerts, A.T.M., (2000), "Mediating Negotiations in a Virtual Enterprise via Mobile Agents", AIWORC2000, Buffalo, NY.

[Wag98] Wagner, G., (1998), Foundations of Knowledge Systems with Applications to Databases and Agents, Kluwer Academic Publishers.

[Wag00a] Wagner, G., (2000), "Agent-Oriented Analysis and Design of the Organizational Information Systems", Proc. of Fourth IEEE Intl. Workshop on Databases and Information Systems, May 2000, Vilnius, Lithuania.

[Wag00b] Wagner, G., (2000), "The Agent-Object-Relationship Meta-Model: Towards a Unified Conceptual View of State and Dynamics", Technical Report, Eindhoven Univ. of Technology, October 2000, Eindhoven,
http://tmitwww.tm.tue.nl/staff/gwagner/AOR.pdf