Realising RosettaNet PIP Compositions as Web Service Orchestrations - A Case Study

Andreas Schönberger
Distributed and Mobile Systems Group
Otto-Friedrich-University Bamberg
Feldkirchenstr. 21, 96052 Bamberg, Germany
andreas.schoenberger@wiai.uni-bamberg.de

Guido Wirtz
Distributed and Mobile Systems Group
Otto-Friedrich-University Bamberg
Feldkirchenstr. 21, 96052 Bamberg, Germany
guido.wirtz@wiai.uni-bamberg.de

Abstract—*A world of ever growing competition not only forces enterprises to continuously optimise their private business processes but also to integrate their business processes with their business partners along the supply chain. RosettaNet is a non-profit standards organisation dedicated to supporting B2B integration. The basis for integrating business processes with RosettaNet are so-called Partner Interface Processes (PIPs) that describe in detail the exchange of business documents for various purposes. We have investigated models for the composition of PIPs for more complex business interactions. The implementation of these models frequently demands for truly distributed computation because central technical infrastructure is either not available or prohibited by business politics.*
This paper is dedicated to the distributed implementation of RosettaNet PIP compositions. We propose Web Services as enabling technique for implementation and WSBPEL for orchestrating Web Service calls. Specifying the correct types of calls in the correct order is a challenging task, particularly if the collaboration itself is complex and is to be executed using insecure communication media. This paper therefore proposes a roadmap for realising RosettaNet PIP compositions as robust and flexible Web Service orchestrations.

Keywords: B2B interaction protocols, business process modelling, WSBPEL, RosettaNet, SOA.

1. Introduction

Enterprises today are enforced by market pressure to integrate business processes with their partners along the supply chain. The term *business collaboration* is used in this paper to refer to the automated interconnection of information systems of business partners for the purpose of business process integration. Building business collaborations gives rise to challenging problems. Personnel from different enterprises with different vocabulary and background are frequently involved in building business collaborations which requires extensive communication support. Central technical infrastructure is frequently not available or prohibited by business politics so that truly distributed computing is needed which is complex by nature. Finally, goods of considerable value may be exchanged during business collaborations which demands for robustness.

To address these challenges, we propose a two-step modelling approach (cf. [1], [2]) that separates business logic, modelled in the so-called centralised perspective (CP), from its distributed implementation, modelled in the so-called distributed perspective (DP). The separation of these perspectives enables business people to concentrate on business issues and to solve communication problems from the CP whereas technical staff can concentrate on distribution issues and care about robustness from the DP.

While this approach can be applied to various application domains, we particularly applied it to RosettaNet PIP ([3]) compositions in a case study. The standards of RosettaNet suit well as the subject of our case study. A large part of the RosettaNet specifications is devoted to standardising message contents of business collaborations, an important task that is not addressed by our approach. Moreover, the RosettaNet standards do not yet contain a proposal for composing PIPs.

This paper is dedicated to the distributed implementation of RosettaNet PIP compositions by means of WSBPEL ([4]) Web Service orchestrations. It identifies important properties of the implementation environment and shows how core challenges can be met.

2. RosettaNet and RosettaNet PIP Compositions

RosettaNet is a non-profit standards organisation dedicated to supporting B2B integration and endorsed by over 500 companies worldwide. Founded in 1998, RosettaNet defines business messages and rules for its electronic exchange. To do so, RosettaNet uses technology and ideas from Open-edi ([5]), UN/CEFACT Modeling Methodology (UMM, [6]) and ebXML[1].

The core RosettaNet standards are Partner Interface Processes (PIPs) and the RosettaNet Implementation Framework (RNIF [3], [7]). PIPs, classified in clusters like cluster *3 Order Management* and segments like segment *3A Quote and Order Entry*, describe the application context, the content and the parameters for the electronic exchange of one or two business documents. The RNIF

[1] http://www.ebxml.org/

published in: Proc. of EEE'06, Las Vegas, June 26th-29th 2006, CSREA Press 2006

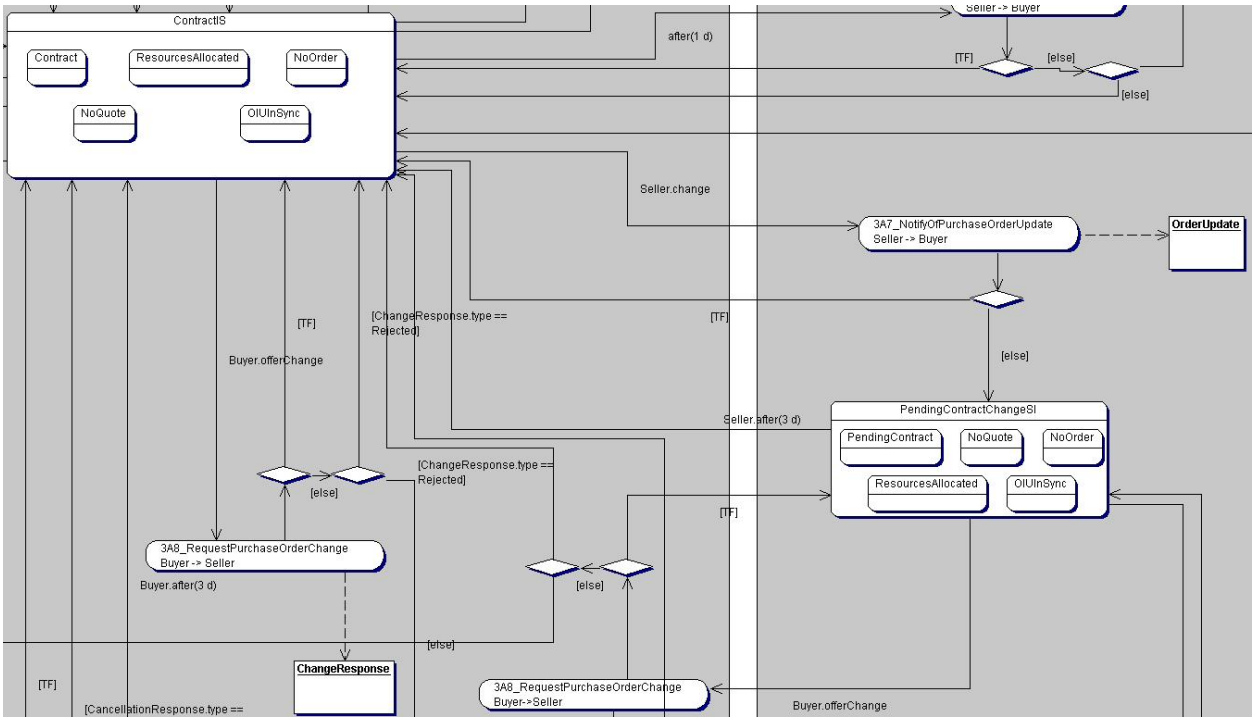

Fig. 1. Extract of a CP model

in turn provides a metamodel for PIPs and details the technology for their execution.

PIPs describe the exchange of one or two business documents at three levels, namely the *Business Operational View* (BOV), the *Functional Service View* (FSV) and the *Implementation Framework View* (IFV).

The BOV describes a PIP from a business perspective. This includes an informal textual description of the application context of the PIP and an UML activity diagram ([8]) visualising the PIP. In that diagram the roles of the business partners involved are represented by swimlanes and an activity node is inserted for every business document to be exchanged. The first activity of such a diagram is stereotyped with a *Business Transaction Type* according to UMM ([6] chapter 1, p.14 f.) and the business documents to be exchanged are visualised as object flows. Finally, the BOV specifies start and end states of a PIP execution and *Business Process Activity Controls* like *Time to Perform* for the overall PIP. Note that the BOV activity diagrams model one single PIP and not a composition of multiple PIPs like [1].

For each role of the BOV a component is defined in the FSV that is responsible for exchanging business documents as *Actions* and control messages as *Signals*. The exchange of each message is detailed by *Message Exchange Controls* and finally the intended order of message exchanges is represented by an UML sequence diagram.

The main task of the IFV is the detailed specification of the business documents to be exchanged which is done in a XSD-file[2]. Moreover the IFV specifies encryption details of the messages to be exchanged.

Apart from defining a metamodel for PIPs, a major part of the RNIF is devoted to specifying a protocol for exchanging the messages of a PIP. This is different from the information in the FSV where only the idealised flow of messages is given. RNIF defines four variants of message exchange protocols as *Business Message Patterns* ([3] p.75 ff) according to which business messages and control messages are exchanged and that can be used to type a PIP. The asynchronous Business Message Patterns, which are predominant among RosettaNet PIPs, are extended in [1] by the well-known 2PC for reliability reasons. The composition of PIPs, not yet addressed in RosettaNet standards, can be modelled in a two-step approach that separates business logic, modelled in the so-called centralised perspective (CP), from its distributed implementation, modelled in the so-called distributed perspective (DP).

Put short, the CP represents a model of the abstract business state of the whole collaboration and how to change that business state. The CP bases on the idea that a business collaboration can be interpreted as a single business process that spans multiple enterprises and that such a business process should be modelled by common states and common state changes. We define these common states as *the common view of the collaboration participants on the collaboration progress* (process state in the following). A process state is composed of the relevant attributes of the collaboration, e.g. the information if a contract has already been signed or if certain resources are free or not. The collaboration participants always reside in the same process state or

[2] `http://www.w3.org/XML/Schema`

if any participant has changed its state then all other participants must make a change to the same state in finite time and no further state changes are allowed until all participants have reached that state. Such a semantics can be achieved by using distributed consensus mechanisms. An alternative approach is to let the views of the collaboration participants on progress diverge but then the number of process states to model would possibly grow exponentially. That is why the use of so-called transactional PIPs is proposed to consistently change process states. The use of participant-local events that are then communicated to the collaboration partners is proposed in order to trigger transactional PIPs. If no communication is possible at all, the use of so-called distributed time-outs is envisaged to make state changes, e.g. for releasing valuable resources of a participant. A distributed time-out is only allowed if communication was successful beforehand, e.g. it can be agreed upon the reservation time of a resource reservation while agreeing upon the reservation itself. Figure 1 shows an extract of the CP of our case study. Process State *ContractIS* represents a state with a valid contract and its relevant attributes as substates. When the *seller* role of the collaboration detects the need for a change, it triggers PIP *3A7 Notify of Purchase Order Update* which leads to process state *PendingContractChangeSI* representing that the *buyer* role of the collaboration still has to decide upon a request for contract change. If the buyer does not trigger PIP *3A8 Request Purchase Order Change* to propagate his decision within 3 days, the collaboration participants switch back to process *ContractIS* because of a distributed time-out.

The CP serves as a means for communication for business analysts in order to determine which process states are relevant for a collaboration, which transactional PIPs can be triggered in a particular process state by which event and to which process states a particular result of a transactional PIP should lead. The CP also serves as the basis for formal verification. More details can be found in [1]. The following sections present how a distributed implementation of such a CP model of RosettaNet PIP compositions can be built.

3. Implementation environment

The core challenge for the implementation of business collaborations is the lack of central technical infrastructure, typically prohibited by business politics. Thus, distributed computing has to be applied in order to connect the information systems of the collaborating partners. This rises the question what quality of the communication media used can be assumed. As small and medium sized enterprises, not being able to spend huge investments on communication facilities, may want to participate in business collaborations as well, an insecure communication media is assumed. Thus, the following constraints have to be addressed:

- There are no assumptions about how long a message travels from sender to receiver.
- Messages can overtake each other.
- Messages can be lost or be duplicated.
- Finally the clocks of the collaboration participants cannot be assumed to be synchronised.

Clearly, these constraints must not prevent the collaboration participants from consistently changing process states because business collaborations may exchange goods of considerable value, i.e. the interaction must handle errors provoked by the communication media like lost messages. One way to achieve this goal is the application of distributed consensus mechanisms that ensure that both collaboration participants agree upon which business messages have been exchanged and what the content of these messages was. Regarding the content of business messages it is clear that its interpretability must be tested before distributed consensus is achieved. This implicitly adds the requirement to the distributed implementation that a business message must be interpretable again (after distributed consensus) in order to perform changes to the real world.

When building RosettaNet business collaborations, systems that cover the necessary business logic are likely to exist already. This concerns the generation and interpretation of business documents as well as the modification of the *real world* and detection of events that trigger PIPs. As considerable investments are likely to already have been spent on these systems, these systems should be reused. Web Services fit well

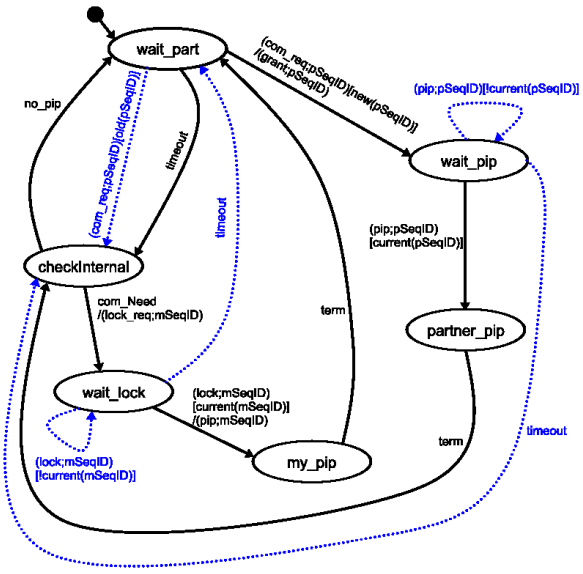

Fig. 2. Coordinator automaton of the MCP

for providing the distributed implementation of a RosettaNet business collaboration. Web Services are platform and programming language independent which is an advantage in integrating heterogeneous systems that are likely to be found when building business collaborations. Further, Web Services are a standard just like RosettaNet deliverables. Finally, Web Services support the message passing paradigm, e.g. with the *In-Only* and *Out-Only* message exchange patterns, which typically is assumed for distributed consensus protocols. In order to build

business collaborations, multiple Web Service calls have to be orchestrated for each participant. WSBPEL, again a standard, can be used to specify the correct types and the correct order of Web Service calls for each participant. Clearly, using an orchestration language like WSBPEL alone does not provide for robust handling of the insufficiencies of the communication media nor does it address the need for integrating existing systems. The next section describes how these tasks can be met.

4. Implementation Roadmap

Relationships between enterprises are highly dynamic, i.e. existing relationships are changed or terminated and new relationships are acquired. Business collaborations therefore must be designed such that they can deal with that highly dynamic setting. The following subsections describe how the main tasks in implementing RosettaNet business collaborations can be met having dynamic relationships in mind.

4.1. Handling communication over insecure media

There are two tasks that need communication among collaboration participants during a business collaboration, i.e. *triggering the execution of PIPs* and *executing PIPs*. The core challenge in implementing these tasks are the assumptions about the communication media in section 3. This challenge can be overcome by using protocols. Fortunately, these protocols can be designed in a generic way and thus reused in different RosettaNet business collaborations. We found it beneficial to use finite automata to specify these protocols. Finite automata did not only simplify reasoning about the protocols but also provided the basis for model checking the protocols with SPIN[3] (cf. [1]). These protocols can easily be mapped to WSBPEL by using a simple enumeration variable for the protocol states to switch over and using WSBPEL *invoke* and *receive* tags for the transitions.

a) Triggering the execution of PIPs: A PIP is usually triggered by the collaboration participant who sends the first message of the PIP. In some process states there may be multiple PIPs that can be triggered, maybe by different collaboration participants. For example, in state *ContractIS* of figure 1 the *Buyer* role could trigger PIP *3A8* whereas the *Seller* role could trigger PIP *3A7*. As the execution of a PIP may lead to new process states where other PIPs can be triggered, the participant with the privilege to trigger a PIP must be determined by a protocol. We have designed the so-called *media control protocol* (MCP) to solve this task. Its goal is, that *at any point in time, at most one collaboration participant has the right to trigger a PIP*. In order to acquire the right to trigger a PIP, collaboration participants first have to request their communication right. If they are granted the right to trigger a PIP and they didn't change their MCP state in the meantime, then they can send the first message of the PIP. Outdated messages of a MCP run

[3] http://spinroot.com

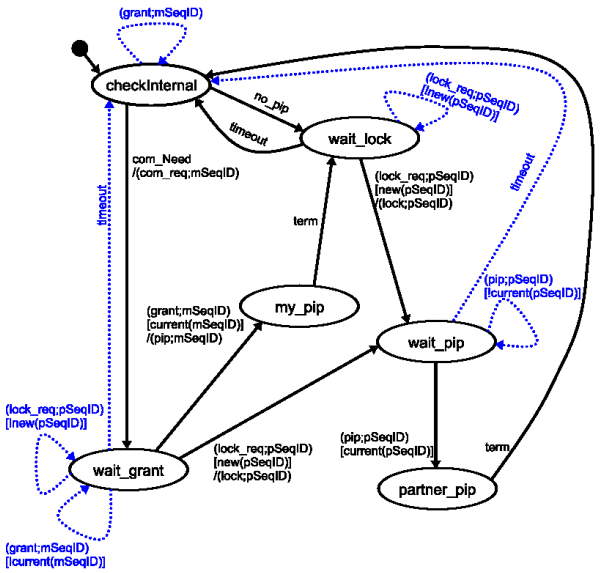

Fig. 3. Participant automaton of the MCP

are handled by means of sequence ids. The situation of concurrent requests for the communication right is solved by an asymmetry due to unfairness. Therefore the roles of *Coordinator* and *Participant* are defined for the MCP with the *Participant* being the disadvantaged role. Figure 2 and 3 show the automata of the MCP. Unfairness is manifest in state *wait_grant* of the *Participant*. The MCP *Participant* waits in *wait_grant* for a *grant* message that represents the right to trigger a PIP, but he can also be interrupted by a *lock_req* message of the MCP *Coordinator* forcing him to let the *Coordinator* go first.

b) Executing PIPs: The RNIF provides *Business Message Patterns* (cf. section 2) for the execution of PIPs, but these lead to rare cases in which diverging views of the collaboration participants are possible. That is why we propose the use of the so-called PIP execution protocol (PIPXP[4] in the following) for ensuring a consistent outcome of PIP executions. Basically the PIPXP is an extension of RosettaNet Business Message Pattern by 2PC. The coordinator of the 2PC run can be determined by choosing the participant who has received the last business document. Typically, all business messages then successfully have been exchanged and interpreted and the only task of the 2PC is to technically agree upon that. The result of such a 2PC run then always will be *Commit*, except communication failures prevent the participants from concluding such a result. In order to avoid blocking processes, manual intervention is needed in the typical *2PC blocking situation*. The notification of such a blocking situation then must be performed reliably which can be safely done because the notification can be performed locally. Figure 4 and 5 show the automata of an Asynchronous Two-Action Activity, i.e. an asyn-

[4] actually, there are slightly different versions of PIPXP according to different RosettaNet Business Message Pattern

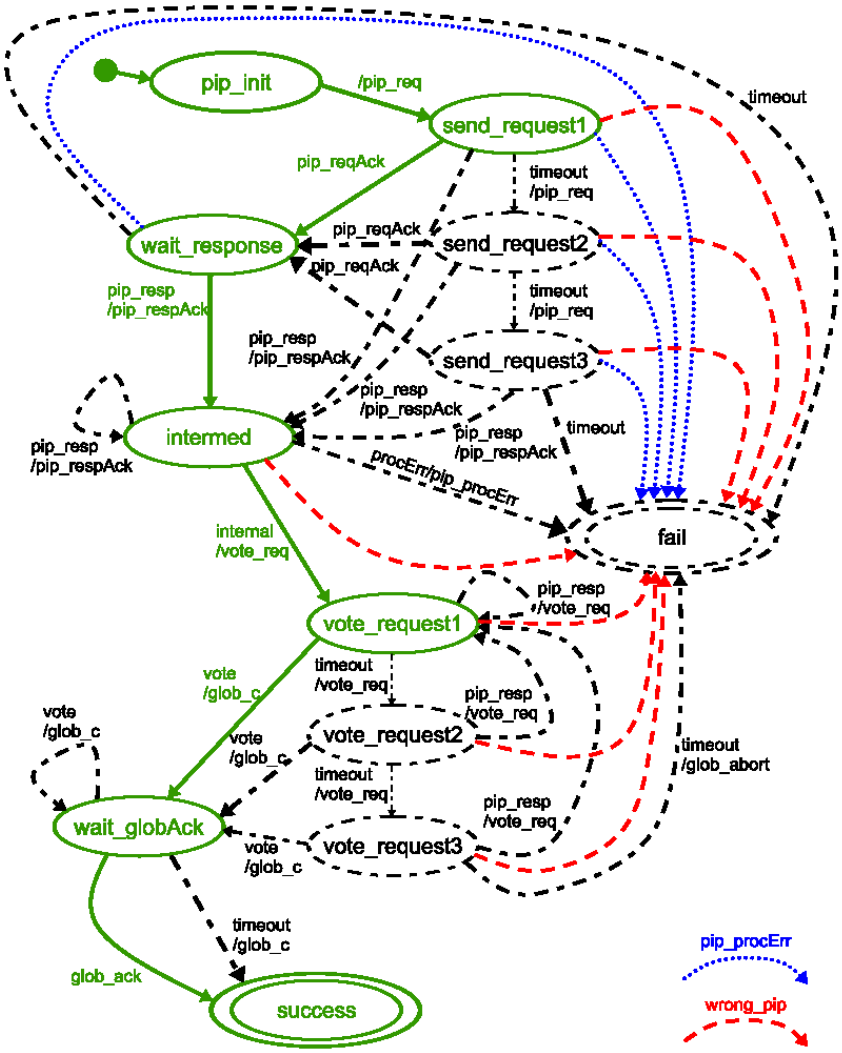

Fig. 4. Sender automaton of a Two-Action Activity

chronous PIP that exchanges two business documents. The sender of the Two-Action Activity is the coordinator of the 2PC run because he receives the last business document. Consequently, the receiver of the Two-Action Activity (the 2PC participant) might be blocked (hang) if *glob_abort* or *glob_c* messages get lost. In this case the transition annotated with *timeout/emitHang* represents any reliable mechanism for notifying a human about the blocking situation.

4.2. Encapsulating business logic

Any business collaboration needs business logic to be executed. Typically, this business logic is closely related to the purpose of a particular collaboration. In order to reuse implementation as far as possible, application dependent business logic must be decoupled from the process that executes the MCP and the PIPXP (protocol process in the following). The tasks in building a RosettaNet business collaboration that need business logic to be fulfilled are the generation and interpretation of business documents as well as the detection of events of the real world (that trigger PIPs) and changing the real world (the implementation of these tasks is called *internal process* in the following).

Decoupling the protocol process from the internal process is not only advantageous because the MCP and the PIPXP can be used in a generic way in the protocol process but also because there already might be systems that are capable of fulfilling the tasks of the internal process. At least, there must be personnel who knows the business logic. Again, Web Services and WSBPEL fit well for encapsulating business logic. As Web Services can be implemented on any platform and with various programming languages, it is easy to provide a wrapper for existing systems or to integrate with a workflow system to interact with personnel. Figure 6 shows how the protocol process can be decoupled from the internal process. As the interaction between protocol process and internal process is local to a collaboration participant, it is assumed to be reliable, i.e. message losses,

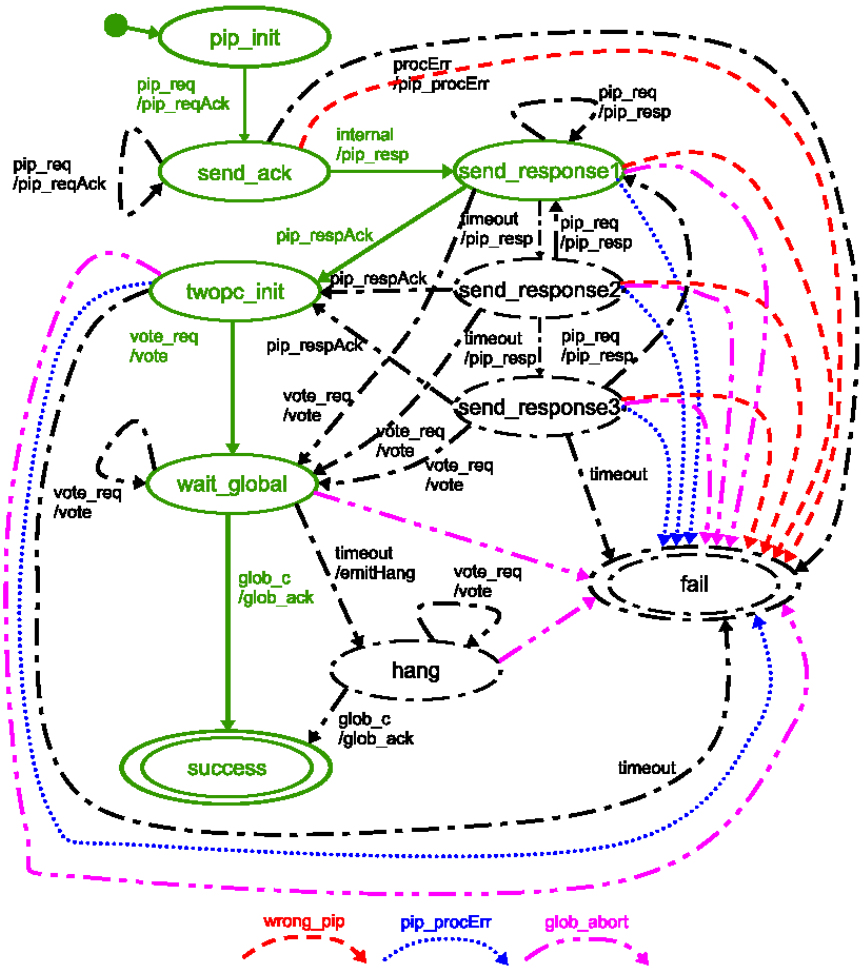

Fig. 5. Receiver automaton of a Two-Action Activity

message duplicates and overtaking messages related to this interface do not have to be explicitly dealt with. The calls between protocol process and internal process are presented with solid arrows for asynchronous calls and dotted arrows for synchronous calls. The arrows always point to the receiver of the call. The calls are used, among other things, to state the need for PIPs (message 1), to announce the possibility for execution of PIPs (message 11), to inform about the result of PIPs (messages 8, 9) or to request for resource reservations (message 5). Yet, in order to implement a RosettaNet business collaboration, the protocol process has to be extended with a representation of the process states and the control flow between process states and PIPs. These tasks also constitute some kind of application dependent business logic. But as opposed to the tasks of the internal process, the information for the implementation of these tasks can be completely derived from the CP on the collaboration and the structure of the PIPs (Business Message Patterns) used. This fact forms the basis for automatic generation of the protocol process. Automatic generation of the protocol process follows the ideas of the OMG MDA[5] approach and thus is a means to handle dynamic relationships between enterprises. Further, automatic generation provides for conformance between the CP and the DP on a business collaboration. Regarding a WSBPEL implementation of the protocol process, process states can be represented by a variable of an enumeration type that contains all possible process states. The control flow can then be realised by a global loop that switches over that variable and applies MCP and PIPXP code according to the CP and the interface for encapsulating the internal process. More details can be found in [1].

5. Related work

Looking at RosettaNet as our use case, [9] have proposed a framework for executing multiple PIPs, but they did not define a modelling approach for creating PIP compositions.

[10] present a mapping from UMM ([6]) concepts *business transaction* and *business collaboration* to WS-BPEL. As RosettaNet and UMM are closely related con-

[5] http://www.omg.org/mda/

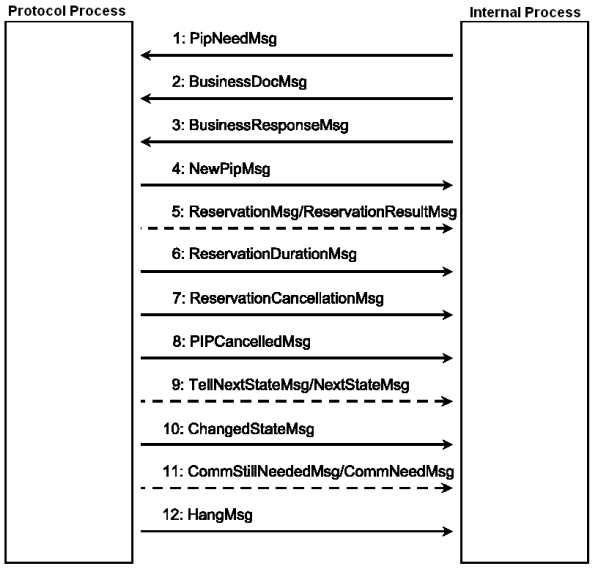

Fig. 6. Message exchange between protocol process and internal process

cerning these concepts the work in [10] is similar to what we do. But the model used for defining *business collaborations* is not as elaborated as the one we presented in [1]. Apart from that [10] do not provide protocols for reliable information exchange nor do they define an interface for using existing systems that implement logic for business document manipulation and detecting events of the real world as well as changing the real world.

In [11] WSBPEL stubs are generated from WS-CDL ([12]) choreographies. This is similar to our work in the sense of deriving a distributed implementation from a global choreography. But [11] provide rules for mapping one standard to another as opposed to the work presented in this paper that is embedded in an approach for modelling business collaborations.

A lot of related work is done by the Web Services community with respect to composing services (e.g. [13], [14]). The composition of services definitely is necessary for integrating businesses but the approaches (excluding WS-CDL) we know all act on the level of single service calls and not on the level of transactional micro-choreographies as we do.

Regarding the fact, that we build a distributed implementation according to a context (i.e. the CP in our case), WS-CAF [15] is similar to what we do but the definition of context is left unspecified.

6. Conclusion and future work

This paper proposes a roadmap for a distributed implementation of RosettaNet PIP compositions that does not need a reliable messaging infrastructure nor synchronised clocks. The roadmap cares for the dynamic nature of business collaborations by reusing the implementation of communication protocols, encapsulating existing systems and deriving information from the centralised perspective of a two-step modelling approach.

Future work is needed for analysing the details of how local business politics of collaboration partners interfere with business properties in the centralised perspective. This knowledge is a must for optimising private processes. Moreover the viability of our modelling approach is to be investigated for other application domains than RosettaNet PIP compositions. Finally, researching the strengths and weaknesses of different technologies in providing a distributed implementation for the CP is an interesting area of ongoing work.

References

[1] A. Schönberger, "Modelling and Validating Business Collaborations: A Case Study on RosettaNet," Otto-Friedrich-Universität Bamberg, Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik 65, Mar. 2006. [Online]. Available: http://www.opus-bayern.de/uni-bamberg/volltexte/2006/81/pdf/modelFIN.pdf

[2] A. Schönberger and G. Wirtz, "Using Webservice Choreography and Orchestration Perspectives to Model and Evaluate B2B Interactions," in *The 2006 International Conference on Software Engineering Research and Practice (SERP'06)*, June 26-29 2006.

[3] *RosettaNet Implementation Framework: Core Specification*, V02.00.01 ed., RosettaNet, www.rosettanet.org, March 2002. [Online]. Available: www.rosettanet.org

[4] IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, *Business Process Execution Language for Web Services*, 1st ed., May 2003. [Online]. Available: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[5] ISO/IEC, *Information technology - Open-edi reference model*, 2nd ed., ISO/IEC, May 2004. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c037354_ISO_IEC_14662_2004(E).zip

[6] UN/CEFACT, "UN/CEFACT' s Modelling Methodology N090 Revision 10," November 2001. [Online]. Available: http://www.untmg.org/

[7] S. Damodaran, "B2B integration over the Internet with XML: RosettaNet successes and challenges," in *Proc. of the 13th international World Wide Web conference on Alternate track papers & posters*. New York: ACM Press, 2004, pp. 188–195.

[8] OMG, *OMG Unified Modeling Language Specification*, 1st ed., Object Management Group, Inc., 250 First Ave. Suite 100 Needham, MA 02494, U.S.A., March 2003. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/03-03-01

[9] A. Dogac et. al., "An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs," ACM SIGMOD Internat.l Conference on Management of Data, 2002.

[10] B. Hofreiter and C. Huemer, "Transforming UMM Business Collaboration Models to BPEL," *Proceedings of the OTM Workshop on Modeling Inter-Organizational Systems (MIOS 2004)*, October 2004. [Online]. Available: http://www.ifs.univie.ac.at/~bh/publications/CoopIS-WS-MIOS-2004-final.pdf

[11] J. Mendling and M. Hafner, "From Inter-OrganizationalWorkflows to Process Execution: Generating BPEL from WS-CDL," *Proceedings of OTM 2005 Workshops*, vol. LNCS 3762, pp. 506–515, 2005. [Online]. Available: http://wi.wu-wien.ac.at/home/mendling/publications/TR05-WSCDL.pdf

[12] W3C, *Web Services Choreography Description Language*, 1st ed., W3C, November 2005. [Online]. Available: http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/

[13] D. Beyer, A. Chakrabarti, and T. Henzinger, "Web service interfaces," in *Proc. of the 14th internat. conference on World Wide Web*. ACM Press, 2005, pp. 148–159.

[14] D. Skogan and R. Gronmo et. al., "Web Service Composition in UML," in *Proc. of the Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04)*, 2004, pp. 47–57.

[15] OASIS Open, "Web Services Composite Application Framework (WS-CAF)," 2003. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf