Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

Composition of negotiation protocols for E-commerce applications

N.K. Diakov, Z.V. Zlatev, S.V. Pokraev

REPORT SEN-R0501 JANUARY 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

Composition of negotiation protocols for E-commerce applications

ABSTRACT
A company doing e-business needs capabilities to negotiate electronically the parameters of its deals in order to fully utilize the potential of the Information and Communication technology (ICT). Businesses continuously engage in interactions of competitive or cooperative nature. This paper focuses on the specification and the composition of negotiation protocols. In particular, we specify and implement a cooperative alliance protocol and we compose it with an auction protocol. We require a specification and implementation language that allows the following two properties of negotiation protocols: (a) compositional construction and (b) dynamic reconfiguration. We apply the Reo coordination language to demonstrate that one can implement and compose together negotiation protocols possessing the desired properties.

1998 ACM Computing Classification System: C.2.4, D.1.3, D.3.2, D.3.3, F.3, J.4
Keywords and Phrases: Automated negotiation, negotiation protocols, coordination, Reo language

Composition of Negotiation Protocols for E-Commerce Applications

Nikolay Diakov
Centrum voor Wiskunde en Informatica
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Tel.: +31 (20) 592 4073
nikolay.diakov@cwi.nl

Zlatko Zlatev
Centre for Telematics and Information Technology
University of Twente, P.O. Box 217 7500AE, Enschede, The Netherlands
Tel.: +31 (53) 489 5334
Z.V.Zlatev@ewi.utwente.nl

Stanislav Pokraev
Telematica Instituut
P.O. Box 589, 7500 AN Enschede, The Netherlands
Tel.: +31 (53) 485 0490
stanislav.pokraev@telin.nl

Abstract

A company doing e-business needs capabilities to negotiate electronically the parameters of its deals in order to fully utilize the potential of the Information and Communication technology (ICT). Businesses continuously engage in interactions of competitive or cooperative nature. This paper focuses on the specification and the composition of negotiation protocols. In particular, we specify and implement a cooperative alliance protocol and we compose it with an auction protocol. We require a specification and implementation language that allows the following two properties of negotiation protocols: (a) compositional construction and (b) dynamic reconfiguration. We apply the Reo coordination language to demonstrate that one can implement and compose together negotiation protocols possessing the desired properties.

1. Introduction

As a result of the diffusion of Internet technology in the mid-90s, the business world encountered a new disruptive possibility [7][9] to exchange data by computer networks at low cost. Like any disruptive technology, computer-based networking has changed business activities and constellations of businesses significantly. The changes in the fundamental equations of business models require rethinking of these models.

One of the business activities that require such rethinking is the coordination among business partners. In the analysis of the potential impact of ICT on business, Malone, Yates and Benjamin [18] predict that in an e-business environment more transactions will be executed through markets than among business units within one company. Not assessing the validity of their prediction, new types of businesses based on execution of market transactions have appeared and sustained; eBay is a good example of such an innovative business. Traditional value nets were disrupted, which led to their deconstruction [11](page 39-69). New formations of cooperating businesses are taking their places [26](page 87-237).

In markets, business activities are coordinated through price, which is the value a business assigns to a resource. Since various businesses assign different values to resources, they need to negotiate to reach mutually acceptable agreements. For this reason, we consider it important to enable businesses to do negotiation in an e-business environment.

A market is only one extreme in the coordination mechanisms used by businesses [18]. A full spectrum of collaborations exists between a spot market transaction and a transaction within one company. It includes various types of alliances differing in the level of interdependences of involved businesses [20][13]. Examples of alliances are: outsourcing partnership, joint teaming relationship, franchise alliance, strategic alliance and joint venture. For this reason, we consider it important to enable businesses to form alliances in an e-business environment.

The coordination mechanisms in markets and alliances require two different types of negotiation: competitive and cooperative, respectively. In this paper, we combine the two negotiation types by composing protocol specifications of two distinct negotiations: an auction (competitive) and an alliance (cooperative).

Our motivating case is an open and dynamic negotiation environment: one with varying number of participants of various types who may join and leave at arbitrary time during negotiation. An example of such a negotiation is an on-line auction, where: (1) any combination of sellers, auctioneers or bidders is allowed and (2) various formations among participants are possible.

We identify [22][26] two problems with negotiation protocols in an open electronic negotiation environment:

1. *Lack of support for temporary business constellations*. In a many-to-many negotiation, participants can form alliances. These alliances are not stable because the shared interests are only temporary; they can break in the course of the negotiation process. Moreover, participants may leave and join the negotiation process at an arbitrary moment in time. Supporting a negotiation protocol with these characteristics requires its implementation to have the ability to adapt to the changes in the negotiation environment;
2. *Inability to deal with nested negotiations*. Apart from the negotiation among alliances, there is a negotiation of similar complexity when forming an alliance. Moreover, negotiation is required within an alliance to prepare each new agreement proposal. The alliance formation and proposal preparation represent separate negotiation processes. Nested negotiations require the ability to compose one negotiation protocol with other ones in a systematic way.

In order to facilitate an open and dynamic negotiation environment, we consider the following requirements for a protocol specification language:

- Dynamic reconfigurability – this allows to accommodate dynamic changes in the negotiation environment, such as changing number or type of participants and forming or dissolving of alliances. This requirement addresses problem 1;
- Composability – this allows to design protocols in a modular style. Furthermore, composability allows to build a negotiation protocol that includes nested independent negotiations. Furthermore, composability during runtime enhances the previous requirement by allowing one to dynamically add new protocols, switch protocols, etc. This requirement addresses problems 1 and 2.

In this paper, we apply the Reo coordination language [5] to demonstrate that one can specify, implement and compose together negotiation protocols that possess the above-mentioned properties. Reo has formal semantics, but also a *formal computational model* that defines the rules according to which one can execute a particular specification.

The rest of the paper has the following organization. We introduce the Reo coordination paradigm and language. Then, we present an example alliance protocol, construct its implementation using Reo, and compose it with an existing auction protocol. After that, we discuss our observations on using Reo with negotiation protocols. We overview related work. At the end, we summarize our results and outline future work.

2. The Reo coordination paradigm

Reo presents a paradigm for composition of software components based on the notion of channels. Reo enforces a channel-based coordination model that defines how designers can build complex coordinators, called connectors, out of simpler ones. Application designers can use Reo as a 'glue code' language for compositional construction of connectors that orchestrate the cooperative behavior of component instances in a component-based system [5]. The Reo coordination language provides, among others, the following features:

- Loose coupling among components;
- Support for distribution and mobility of heterogeneous components;
- Exogenous coordination (i.e., by third parties);
- Dynamic reconfigurability that allows one to change a connector during runtime using topological operations;
- Formal semantics based on a coinductive calculus of flow [2][23] and (alternatively) on constraint automata [3].
- Formal computational model that defines the rules for computing Reo connectors in a distributed computing environment [11];
- A serialization of its visual notation in XML validated by XML Schema, for interoperability among design and analysis tools;
- A comprehensive visual notation.

2.1. Basic concepts

From the point of view of Reo, a system consists of a number of *component instances*, interacting through *connectors*. Reo assumes that a component instance contains one or more active entities (e.g., processes, agents, threads, actors, etc.), which communicate with entities outside of their component instance only through connectors. Reo completely abstracts from the details of the communication within a component instance. Instead, Reo focuses on the communication between component-instances, which takes place exclusively through connectors. Reo allows compositional construction of a connector out of simpler connectors, where *channels* represent the *atomic connectors*.

A channel has precisely two *channel ends*. Reo introduces two types of channel ends: *sink* and *source*. A sink dispenses data out of its channel. A source accepts data into its channel.

Reo models a connector as a graph of *nodes* and *edges*, where zero or more channel ends may coincide on every node, every channel end coincides on exactly one node, and an edge exists between two nodes if and only if there exists a channel whose channel ends coincide on those nodes.

Reo has three types of nodes: *mixed*, *source*, and *sink*. A mixed node contains both source and sink channel ends. A mixed node serves as a pumping station for its coincident

channel ends: it non-deterministically selects a data item available at one of its sink channel ends and replicates the data item to all of its source channel ends when all of them can accept the data item. A source node contains only source channel ends. If a component writes a data item to a source node, the node replicates the data item to all of its source channel ends when all of them can accept the data item. A sink node contains only sink channel ends. When a component tries to take a data item from a sink node, the node non-deterministically selects a data item available at one of its sink channel ends.

2.2. Reo operations

Any active entity inside a component instance can perform Reo operations on a channel end. Reo defines two types of operations: topological – ones that allow manipulation of connector topology, and IO – ones that allow input/output of data. Due to space limitation, we discuss only operations that we use later in the text.

Topological operations include, among others, *join* and *split*. The join operation allows joining of two nodes identified by two channel ends, each coincident with one of the nodes. The split operation allows for splitting a node into two nodes by specifying the channel ends that the performer requires to coincide on the new nodes.

IO operations include, among others, *take* and *write*. The take operation allows the performer to read and remove a data item from a sink. The write operation allows the performer to write data to a source.

2.3. A useful set of primitive channels

Reo assumes the availability of an arbitrary set of channel types, each with well-defined behavior. In this paper, we consider the following non-exhaustive set of channel types, each with some distinctive properties: Sync, Filter, SyncDrain, LossySync, FIFO1. A Sync channel has a source and a sink. Writing a message succeeds on the source of a Sync channel if and only if taking of a message succeeds at the same time on its sink. The Filter channel behaves like the Sync except that it loses all data that do not match the specified pattern of the Filter. A SyncDrain has two sources. Writing a message succeeds on one of the sources of a SyncDrain channel if and only if writing a message succeeds on the other source. A LossySync channel has a sink and a source. The source always accepts all data items. If the sink does not have a pending read or take operation, the LossySync loses the data item, otherwise the channel behaves as a Sync. The Sync, Filter, SyncDrain, SyncSpout and LossySync belong to the family of synchronous channels. The FIFO1 channel has a source and a sink. The FIFO1 channel maintains a buffer with capacity of one. Writing a message succeeds on the source of a FIFO1 if and only if its buffer does not contain any messages. Taking of a message succeeds on the sink of a FIFO if and only if its buffer already contains a message. The FIFO1 belongs to the family of asynchronous channels.

Figure 1 shows the visual notation for the basic channels we introduced above.

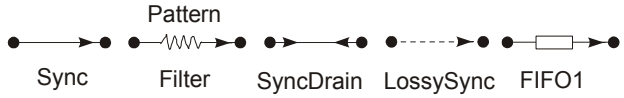

Figure 1. Visual notation for basic channels

2.4. Connector encapsulation

In analogy with electrical circuits, we call a design a *circuit* in Reo. When designing large circuits, we find it useful to abstract from the details of a particular connector that we want to instantiate and reuse. In Reo, we do this through encapsulating the circuit of a connector. In the visual notation we represent an encapsulated circuit using a box. On the borders of the box, we position the nodes that the connector exposes to the outside. In effect, the box represents a new component, with internal behavior defined entirely in Reo.

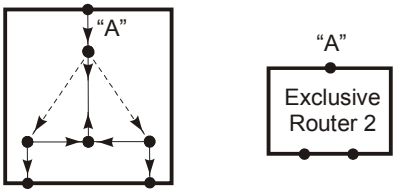

Figure 2. Exclusive Router 2 connector (left) and one of its instances (right)

Figure 2 shows how we define the Exclusive Router 2 connector [2]. The Exclusive Router 2 routes synchronously its input to precisely one of its outputs. We also show how we depict an instance of the Exclusive Router 2. Note that we may label nodes on the border of a connector, in order to assign some designer meaning to the messages passing through that node. This labeling serves only to simplify the presentation of a circuit, and to allow designers to distinguish the role of the nodes, i.e., input or output. It has no implications on the semantics of the circuit.

3. Example: auction with alliances

In this section, we apply Reo in an example business case that includes negotiations. These require the composition of protocols with the properties listed in section 1.

3.1. Business case

A company called DeRio produces a certain range of products. DeRio has recently closed a deal with a new partner situated at a distant location. To fulfill its contractual obligations, DeRio needs a transportation service from its factory to the customer address. Figure 3 shows a possible transportation route between point A, the factory and point E, the customer address.

DeRio decides to set up an auction to determine the best service offer. All parameters of the services, such as delivery deadline and payment terms, are fixed. The price is the only negotiable parameter. Several companies respond to the announcement; among them are DHS and

UPL, widely recognized players in the transportation domain and Neptun and Aviz, companies specializing in particular kind of transport, e.g. water or land transport.

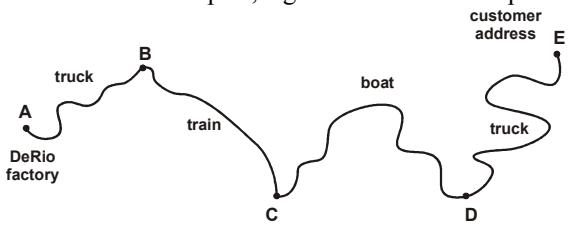

Figure 3. Transportation route

Neptun and Aviz cannot compete individually with DHS and UPL for some reason (e.g., Neptun only offers boat and train transport at good prices, while Aviz offers only truck transport at good prices). To gain competitive advantage, they decide to form an alliance called AlNeviz. Figure 4 shows a schematic setup of the auction, the participants and the alliance.

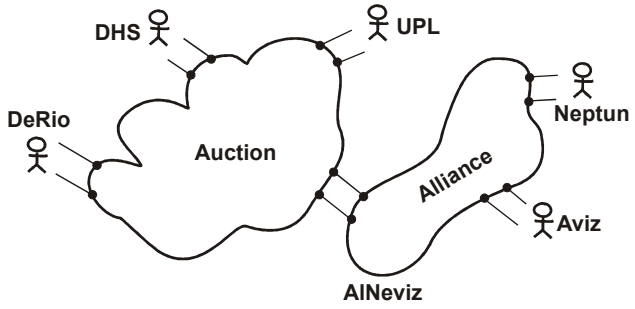

Figure 4. DeRio auction setup

Our business case includes two negotiations, namely: a negotiation in a form of auctioning among bidders and a negotiation among partners within an alliance. We covered the auctioning in previous work [26]. Further in this section, we specify and implement the alliance protocol, and compose it together with the auction protocol.

3.2. Alliance protocol

In our example business case, we consider two distinct roles that alliance participants can play:

- Chairman – hosts the alliance. In this example, Neptun;
- Ally – participates in the alliance. In this example, Aviz and Neptun.

Below, we list an informal specification of the alliance protocol. The numbering represents an enumeration; it does not suggest a particular sequence in applying. When we introduce a term for the first time we show it in *italic*:

1. An alliance has one *chairman* and at least one *ally*;
2. Upon alliance initiation all participants are informed about the *current price* and *bid step*;
3. At any moment participants are informed about changes in the auction's current price and bid-step;
4. The next auction bid is determined in two distinct subsequent phases: (1) *bid-determination* phase, and (2) *involvement-discussion* phase;
5. During the bid-determination phase, each participant submits its tender to the chairman. The next bid becomes the highest valid bid of all bids submitted during this phase. Each participant is informed about the potential next bid;
6. During the involvement-discussion phase, participants take turns to discuss their involvement in the next bid. Everyone has an equal opportunity to send its proposals to the other participants;
7. During the involvement-discussion phase, a fall of the current auction price under the potential next bid triggers a new cycle of the bid-determination and involvement-discussion phases;
8. The chair decides when to end the involvement-discussion phase;
9. After the completion of the involvement-discussion phase, the chair submits the next auction bid;
10. Upon closing of the auction, participants are informed about the outcome of the auction.

3.3. Protocol specification and implementation

Using the specification from the previous section, we construct a Reo circuit that specifies the alliance protocol. The resulting specification of alliance circuit also serves as an implementation, because Reo has a formal computational model. First, we introduce basic components; then, an external library of connectors; then, the larger auxiliary connectors; and at the end, we introduce the alliance protocol circuit.

3.3.1. Basic components

In addition to the basic channels introduced earlier, we need several basic components that can operate on the data passing through channels (Figure 5).

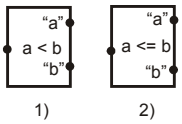

Figure 5. Basic components

The first component offers two source (input) nodes and one sink (output) node. We have labeled the sources with "a" and "b". When we write two messages a and b representing integers to "a" and "b" respectively, the component outputs "true" if $a < b$ and "false" otherwise, through its sink. The second component has the same characteristics as the first one; however, it outputs "true" if $a <= b$ and "false" otherwise. One can find a method for formal definition of the behavior of the components in [5] using the Reo algebraic semantics [23].

3.3.2. Library of connectors

We depict in gray color all component instances used in the connectors and alliance protocol circuit that we refer to [2][26] for their definition. These components constitute: Initially Closed Valve (ICV), Initially Opened Valve (IOV), Exclusive Routers with more than two outputs, Sequencer,

Sequencer with reset, Bid Validator, and Constant Writer. Both valves regulate the flow of data; however, the ICV initially does not allow flow, while IOV initially allows flow. Both valves have nodes through which one can toggle their state from opened to closed and the other way around. An exclusive router of order higher than two routes to more than two channel ends. A Sequencer produces a data item from its outputs in a pre-defined order. A Sequencer with reset allows one to reset the Sequencer to its initial state. A Bid Validator determines the validity of a data item representing the bid in an auction. A Constant Writer produces the same data (determined during instantiation) item each time someone asks to read from its offered channel end.

3.3.3. Larger connectors

Using the basic components, we build three larger connectors that we use in the protocol implementation.

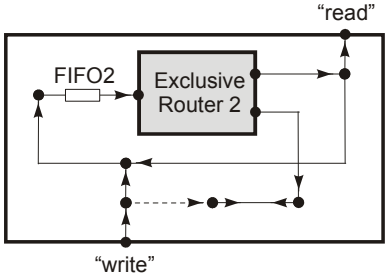

Figure 6. Variable (un-initialized)

The Variable connector (Figure 6) serves as a placeholder for data items similar to a variable in imperative programming languages. It offers a sink node and a source node. The Variable always accepts a message written on its source and the last message written represents the value of the variable. The Variable always offers a message containing its value to anyone that makes a take operation on the component's sink. An un-initialized Variable makes a read on its sink to wait for write to supply an initial value.

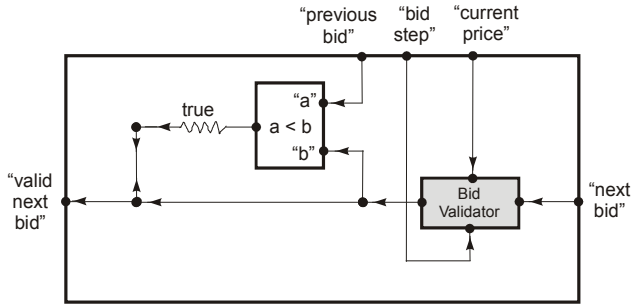

Figure 7. Next Bid Validator

The Next Bid Validator connector offers four source nodes and one sink node. The Next Bid Validator represents a component that takes as input the previous bid, the current price in the auction, the next bid, and the current bid step, and outputs the bid it received as input, if and only if the value of the bid meets the requirements described in rule 5 of the alliance protocol. We use this component to determine the validity of bids made by bidders, in order to determine the next bid of the alliance on the auction.

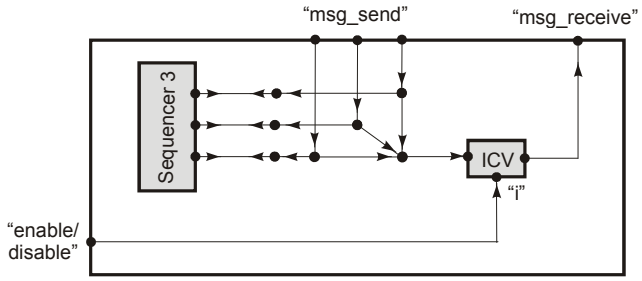

Figure 8. Generic Com System 3 (initially disabled)

The Generic Com System 3 connector offers four source nodes and one sink node. The Generic Com System 3 represents a component that takes as input messages from three of its source nodes and routes these messages to its sink node. We use a Sequencer to enforce an equal opportunity for participants to exchange messages with other participants (rule 6). The Generic Com System 3 initially does not allow sending or receiving of messages. One can enable or disable (toggle) the Generic Com System 3 by writing a signal message to its fourth source node. Furthermore, one can easily modify the Generic Com System 3 to facilitate an arbitrary number of senders, where each sender must exclusively write to only one of the sources. Note that if a participant acts as both sender and receiver, it will receive its own messages too. We use this component in the alliance protocol to allow participants to discuss their involvement in the next bid of the alliance during the involvement-discussion phase.

3.3.4. Alliance protocol circuit

In this section we construct an alliance protocol.

Figure 9 shows the visual specification of the alliance protocol circuit. Using tools, one can obtain an XML serialization of this specification. Furthermore, one can also obtain a complete algebraic specification by applying the Reo algebraic semantic rules for channel composition to all channels and component instances in the circuit.

The Alliance protocol consists of one instance of the Alliance connector, one instance of the Generic Com System N connector, and two or more instances of an Participant connector. Note that N represents the number of participants in the alliance. The Alliance connector coordinates the activities of the chairman – Neptun (that represents the alliance). The Participant connector coordinates the activities of participants – both Neptun and Aviz. Figure 9 depicts only one Participant instance. The Alliance connector automatically registers and pays the auction fee to the auction.

A participant in the alliance interacts with other participants exclusively through an instance of a Participant connector.

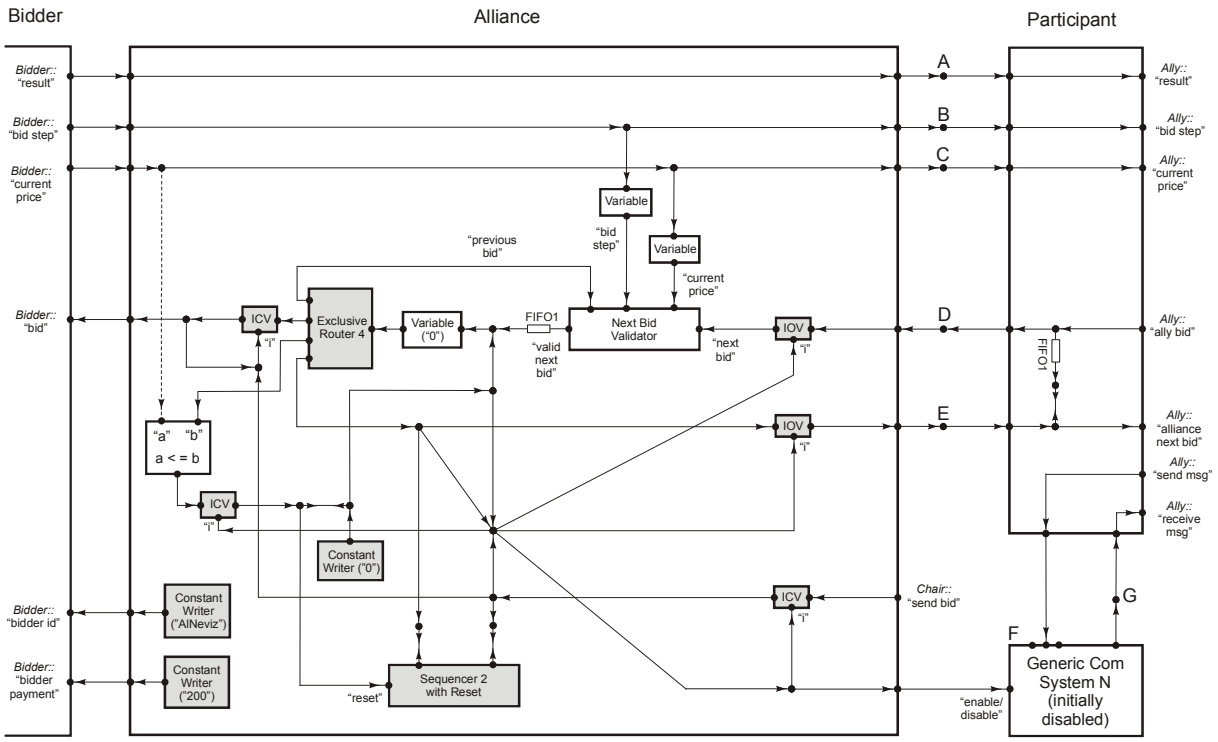

Figure 9. The Alliance protocol circuit

In addition to using a Participant connector instance, the chairman also performs its chairman duties using a labeled node on the Alliance connector.

All Participant instances connect to the Alliance and Generic Com System N through joining their nodes with the respective sink and source nodes A, B, C, D, and F (using auxiliary Sync channels).

In this implementation, we have made several technical choices that the informal protocol description does not specify in detail. When facing such technical choices, we prefer the simpler ones in terms of Reo primitives used to implement the circuit. Furthermore, because we want to keep the implementation comprehensible, we do not check for proper input values nor do we perform error handling.

3.4. Composition of the alliance protocol and the auction protocol

Figure 10 shows the composition of the two negotiation protocols. For each protocol, we only depict one instance of each connector type. Consult the description of the individual protocols and the cardinality of their relations.

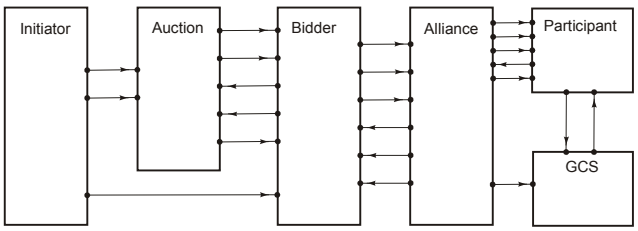

Figure 10. Auction protocol with alliances

The Initiator, Auction and Bidder connectors belong to the auction protocol [26]. The alliance appears to the auction as a normal bidder. The auction protocol provides a separate Bidder connector instance to each of its bidders. Therefore, the alliance also has precisely one Bidder connector instance in order to interact with the auction circuit. We connect an Alliance connector instance to a Bidder connector instance by joining the corresponding input/output nodes of the Bidder to the corresponding nodes of the Alliance circuit with the help of auxiliary Sync channels.

4. Observations on using Reo with negotiation protocols

In this section, we present several observations, we made during working with Reo on the auction and alliance protocols. Furthermore, we give an idea about the additional possibilities that become available to protocol designers, should they decide to select Reo as their specification and implementation language.

The Reo coordination paradigm allows for strict separation of coordination from data processing. Therefore, we can assess the amount of pure coordination in comparison with data manipulation in the alliance protocol. Gray components, the Variable instances, the Generic Com System N, and all channels from Figure 9 (the alliance protocol circuit) represent coordination elements. The white components (except the Variable instances and the Generic Com System N) perform some data processing. One can see the significant amount of coordination that occurs in this

implementation of the alliance protocol. Using a coordination language, such as Reo, which treats coordination as a first class modeling concept, allowed us to deal more efficiently with the coordination issues in the alliance protocol. This observation illustrates that designers can use Reo also for modeling of other e-commerce applications that involve significant amount of coordination.

The strong formal apparatus behind Reo gives the following additional advantages:

- Simulator for Reo – a tool that implements a non-distributed version of the Reo computational model, to allow running and testing protocol prototypes;
- Distributed coordination middleware for Reo – a tool (under development) that implement the full Reo computational model to allow one to run Reo circuits directly in a distributed environment. The Reo coordination middleware allows plugging in of different component models for structuring the user applications into entities, and of transport protocols for the communication between the entities at the level of channels;
- Model checker – Time Scheduled Data Stream Logic [4] represents an existing theory that a model checker tool (under development) can use to check properties, such as, liveness, reachability, and deadlock conditions.

5. Related work

Several coordination languages have been proposed and used for negotiation protocol specification. Two such languages from the Multi-Agent Systems field are AgenTalk [17][16] and COOL [6]. They are based on Finite State Machines (FSM) and used in negotiation protocol specification. Nevertheless, they are limited in their expressiveness and not suitable for the negotiation environment described in section 1. The AgenTalk language does not have formal semantics [17] and does not support dynamic reconfigurability and composability. AgenTalk has an inheritance mechanism as means for reuse of design but this is not dynamic. The COOL language uses FSM as its formal basis, where the transitions are exchanges of speech-act-based messages. COOL does not have formal computational model and does not have language primitives for dynamic reconfiguration.

Another group of specification languages originates from the Web services initiative. One example is the Business Process Execution Language for Web Services (BPEL4WS)[10]. BPEL4WS combines the formalisms used in its predecessors, namely the support for graph oriented processes from WSFL and the structural constructs for processes from XLANG [25]. BPEL4WS is designed for composition of Web services; although, it lack the dynamic reconfigurability. This and other drawbacks are discussed by Kim et al. [15] and Van der Aalst [1].

Web Services Conversation Language (WSCL) [24] is language that specifies the documents exchanged and the sequence the document exchanges. WSCL is limited in its expressiveness; it limits the number of the participants in a conversation to two, does not support parallel activities, and decision activities can only use as conditions the output of the a preceding activity. WSCL does not have language primitives for dynamic reconfiguration of the conversation protocol.

Business Process Modeling Language (BPML) [8] is language similar to XLANG. BPML has a well-defined but not formal semantics. It provides composability, transactions, executable specifications, dynamic participation, etc. However, BPML does not have language primitives for dynamic reconfiguration.

General formal modeling techniques exist, such as π-calculus, data-flow models, Kahn-networks, and Petri-nets. We view them as specialized channel-based models that incorporate certain specific primitive coordination constructs [5]. Recent work [13] on comparing Reo and Petri-nets, showed that one can relatively easy transform existing Petri-net models into a Reo circuit, while the opposite proves to be difficult. In our view, the inherently dynamic topology of connectors and the very liberal notion of channels make Reo more general, and hence more powerful.

6. Conclusions

In this paper we have presented an approach for composition of negotiation protocols using the Reo coordination language. We use the inherent features of Reo, such as composability and dynamic reconfiguration, to produce a specification of an alliance protocol that can facilitate a dynamic and open negotiation environment. We compose the alliance protocol together with an auction protocol to form a new auction with alliances protocol.

Reo offers topological operations that one can use to modify a Reo circuit during runtime. These allow to adding a new bidder to or remove an existing one from the protocol circuit. This addresses the problem of temporary alliances, which may dissolve during the negotiation process should a disagreement arise among the allies. To our knowledge, only specialized channel-based models allow for some specific forms of dynamic reconfigurability.

The Reo composability and dynamic reconfigurability allow to design and to implement protocols in a modular style. In such a way, we isolate and reuse coordination designs. After a composition, the overall behavior of the new negotiation process remains predictable. Reo composability allows for support of nested negotiation protocols. We demonstrate this with assembling an auction protocol together with an alliance protocol. Non-composable languages, such as AgenTalk, COOL,

BPEL4WS, and WSCL, do not allow one to derive properties of a composition from its constituent nested protocols.

7. Future work

We want to deploy and study negotiation protocol implementations with alliances in a distributed environment. This will contribute to the point we make that Reo does not only constitute modeling paradigm, but also an executable paradigm in a distributed environment.

8. References

[1] Aalst van der, W.M.P., Don't go with the flow: Web services composition standards ex-posed, *Trends and Controversies*, IEEE Intelligent Systems, Jan/Feb, 2003.

[2] Arbab, F., Abstract Behavior Types: A Foundation Model for Components and Their Composition, In: *Proceedings of the First International Symposium on Formal Methods for Components and Objects (FMCO 2002)*, LNCS 2852, 2003, pp.33-70.

[3] Arbab, F., Baier, C., Rutten, J., Sirjani, M., Modeling Component Connectors in Reo by Constraint Automata, *Electronic Notes in Theoretical Computer Science*, vol. 97, No. 22, July, 2004, pp. 25-46.

[4] Arbab, F., Baier, C., de Boer, F., Rutten, J., Models and Temporal Logics for Timed Component Connectors, *Proc. of the IEEE International Conference on Software Engineering and Formal Methods (SEFM '04)*, Beijing, China, 26-30 September, 2004.

[5] Arbab, F., Reo: A Channel-based Coordination Model for Component Composition, *Mathematical Structures in Computer Science*, Cambridge University Press, Vol. 14, No. 3, June 2004, pp. 329-366.

[6] Barbuceanu, M. and Fox, M.S., COOL: A language for Describing Coordination in Multi Agent Systems, *Proc. First International Conference on Multi-Agent Systems (ICMAS'95)*, 1995, pp. 17-24.

[7] Bower, J. L. & Christensen, C. M., *Disruptive technologies: catching the wave*, Harvard Business Review, 73, 1 (January-February) , 1995, pp. 43—53.

[8] BPMI.org, BPML 1.0 Specification, online available at: http://www.bpmi.org/specifications.esp

[9] Christensen, C.M., *The Innovator's Dilemma*, Harvard Business School Press, Boston, Mass, 1997.

[10] Curbera, F., Goland, Y.,Klein, J., Leyman, F., Roller, D., Thatte, S. and Weerawarana, S., Business Process Execution Language for Web Services (BPEL4WS) 1.1, May 2003, online available at: http://www.ibm.com/developerworks/library/ws-bpel/

[11] Evans P. & Wurster, T. S., Blown to Bits: How the New Economics of Information Transforms Strategy, Harvard Business School, 2000

[12] Everaars, K., Costa, D., Diakov, N.K., Arbab, F., Reo Rewrite Rules: Functional Specification, work in progress, presented at The Amsterdam Coordination Group (ACG), May, 2004.

[13] Gordon, M., Vantage Partners, LLC, Negotiating the Alliance, A.S.A.P.'s Collaborative Commerce Summit 2001 and Alliance Best Practices Training Workshops, online available at: http://www.strategic-alliances.org/slideshows/Summit2001Scottsdale/GordonVantPart.pdf

[14] Guillen-Scholten, J., A Translation from Reo to Petri Nets and Vice-Versa, work in progress presented at the The Amsterdam Coordination Group (ACG), June, 2004.

[15] Kim, J.B., Segev, A., Patankar, A.K., Cho, M.G., Web Services and BPEL4WS for Dynamic eBusiness Negotiation Processes, In: Zhang, L. (ed.), *Proceedings of the International Conference on Web Services*, ICWS '03, Las Vegas, Nevada, USA. CSREA Press 2003, 2003, pp. 111-117, ISBN 1-892512-49-1.

[16] Kuwabara, K., Ishida, T., and Osato, N., AgenTalk: Describing Multiagent Coordination Protocols with Inheritance, *Proc. 7th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '95)*, 1995, pp. 460-465.

[17] Kuwabara, K., Ishida, T., and Osato, N.,AgenTalk: Coordination Protocol Description for Multiagent Systems, *Proc. First International Conference on Multi-Agent Systems (ICMAS '95)*, 1995, p. 455.

[18] Lam M.D., Why Alliances Fail, Pharmaceutical Executive, 2004, online available at: http://www.pharmexec.com/pharmexec/article/articleDetail.jsp?id=98299&pageID=1

[19] Malone, T.W., Yates, J., Benjamin, R.I., Electronic Markets and Electronic Hierarchies, *Communications of the ACM*, vol. 30, June, 1987, pp. 484-497.

[20] Margulis, M. & Pekár, Jr., P., The Next Wave of Alliance Formations: Forging Successful Partnerships with Emerging and Middle Market Companies, Houlihan Lokey Howard & Zukin, 2004, online available at: http://www.hlhz.com/download.asp?fid=406

[21] Mousavi, M.R., Sirjani, M., Arbab, F., Specification, Simulation, and Verification of Component Connectors in Reo, *Technical Report No. 04-15*, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, June 2004.

[22] Pokraev, St., Zlatev, Z., Brussee, R. & Eck, P. van, Semantic Support for Automated Negotiation with Alliances. In Seruca, E. [et al.] (eds.): *Proceedings of the Sixth International Conference on Enterprise Information Systems*, ICEIS 2004, Porto INSTICC, Portugal, April 14-17, vol.4, 2004, pp. 244-249.

[23] Rutten, J.J.M.M., Kwiatkowska, M., Gethin, N., Parker., D., Chapter 5: Component Connectors, In *Mathematical Techniques for analysing concurrent and probabilistic systems*, CRM Monograph series Volume 23, American Mathematical Society, 2004, p. 215.

[24] W3C, Web Services Conversation Language (WSCL) 1.0, W3C Note, March, 2002, online available at: http://www.w3.org/TR/wscl10/

[25] Weerawarana, S. and Curbera, F. P., Concepts in business processes, 2002, online available at: http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/

[26] Weill, P. & Vitale, M., Place to Space: Migrating to Ebusiness Models, Harvard Business School Press, 2001

[27] Zlatev, Z., Diakov, N. and Pokraev, S., 2004, "Construction of Negotiation Protocols for E-Commerce Applications", ACM SIGecom Exchanges, Vol. 5, No. 2, November, pp. 12-22.