

SPECIFICATION AND IMPLEMENTATION OF MULTI-AGENT ORGANIZATIONS

Fatemeh Ghassemi, Naser Nemat Bakhsh, Behrouz Tork Ladani

Department of Computer Engineering, Isfahan University, Isfahan, Iran
{Fghassemi,nemat,ladani}@eng.ui.ac.ir

Marjan Sirjani

Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
msirjani@ut.ac.ir

Keywords: Organization, Formal Model of Organization.

Abstract: Multi-agent systems are used as a solution for complex and distributed systems. Since agents are autonomous they can be coordinated exogenously by a coordination language Reo. Reo coordinates agents without having any knowledge about agents. We apply organizational concepts to analyze and design such systems. In this paper, we propose a formal model to specify the results achieved during these phases. This formal model helps in designing a coherent and consistent system. The formal model is applied to make the implementation of system by Reo systematically. We will specify and implement system by Reo according to the formal model. This paper also defines how to convert the formal specification to a Reo circuit by providing Reo circuits for the different patterns of interaction protocols and how to compose simpler circuits to support more complex patterns.

1 INTRODUCTION

Autonomous agents and multi-agent systems (MASs) are widely used by developers to design complex and distributed systems such as e-learning and e-marketing systems and business-to-business applications. An agent provides a behaviour abstraction which allows the developers to naturally model and construct complex systems.

In the context of MASs, the autonomous and proactive behaviour of agents suggests that applications can be designed by mimicking the behaviour and structure of human organizations. Thus the architecture of a multi-agent system can be naturally viewed as a computational organization, which consists of a multitude of autonomous and interacting agents. Each agent plays one (or more) specific roles. However, the organization of a multi-agent system is distinct from the individual agents that populate the system (Zambonelli et al., 2000; DeLoach, 2002; DeLoach and Matson, 2004). While agents play roles within the organization, their roles do not constitute the organization. Roughly speaking organizations are characterized by the organizational structures as well as organizational rules that define the requirements for the

instantiation and operation of the organization as well as constraints on agent behaviours and interactions (Zambonelli et al., 2000; DeLoach, 2002).

Organization defines and coordinates agent interactions. So a multi-agent system is defined by a set of agents and its coordination (Dastani et al., 2005). We only consider the external behaviour of agents. Agent organization can be open, where agents enter or leave dynamically. Thus agents are not known to each other and they may not be honest to each other. However there are some problems in the specification and implementation of an open organization as follows:

- Participants in an organization should have a common understanding of the organizational rules and organizational structure. We consider the source of this problem the use of protocol specification languages with poor formal semantics.
- Some of agents in a large organization may make a new sub-organization to cooperate for a specified task. Since agents are autonomous, they are not affected by this organizational changes and it is the responsibility of the organization to manage dynamic changes. Supporting such characteristics for an organization requires implementation to have the ability to adapt changes in organization policies.

- There is always a gap between system specification and implementation.

In this paper, we apply the coordination language Reo to address above problems. Reo has a formal semantics and is dynamically reconfigurable. To implement an organization using Reo language, we propose a formal model to specify the organization of multi-agent systems. Then we use Reo to implement the specification of the organization. Applying the formal specification makes the implementation process by Reo language systematically. We can use the formal semantics of Reo to evaluate the properties of the organization and overall system performance, security, flow of information, etc.

Structure of the paper: The organization metaphor is described in Section 2. In Section 3 we explain the Reo concepts and in Section 4, we explain our formal model for an organization. In Section 5, we explain how to implement an organization by Reo language. In Section, 6, we specify and implement an example system using our formal approach. Finally in Section 7, we explain our concluding remarks and future works.

2 ORGANIZATION

In the traditional design of concurrent and distributed systems, the architecture is derived from the decomposition of functionalities and the data required by system to achieve its goals as well as the definition of their inter-dependencies. However, using organizational concepts to design such systems, leads to a number of agents, each with specific roles in the system. In this model, agents interact to accomplish their tasks and, agents embed most of the functionalities they need, so the interactions of agents are reduced which makes the design less complex and easier to manage. Most MASs are intended to support or control some real-world organizations. In such cases, an organizational-based MAS design reduces the conceptual distance between the software system and the real world system it has to support.

An organizational structure defines the specific class (among the many possibilities) of organization and control regime to which the agents/roles have to conform in order for the whole MAS to work efficiently and according to its specified requirements (Zambonelli et al., 2000). These organizational structures are usually described in terms of a variety of social and organizational concepts such as norm, trust, power, delegation of task, responsibilities, permission, access to resources and communication (Dastani et al., 2005). The organization structure defines admissible actions of agent interactions. For instance when there is delegation

relation between two agents, one agent can delegate task to another agent. So, the delegating agent has a delegation action in its interaction protocol.

Organizational rules express general, global (supra-role) requirement for the proper instantiation and execution of a MAS (Zambonelli et al., 2000; DeLoach, 2002). These rules indicate some constraints between two communicating agents or an agent and organization.

3 REO CONCEPTS

Reo is a channel-based exogenous coordination language based on the calculus of channels (Arbab, 2004; Arbab and Rutten, 2003; Arbab, 2003). Reo consists of components that are connected via complex coordinators, called connectors or networks, which coordinate their activities. Connectors are compositionally built out of simpler ones. The simplest connectors in Reo are a set of channels with well-defined behaviour supplied by the users (Arbab, 2004). Agents communicate with each other by means of I/O operations they perform through the I/O interfaces of the connectors. The connector imposes a specific coordination pattern on agent actions without any knowledge about their internal communications. A channel has precisely two channel ends. There are two types of channel ends: sink and source. A sink channel end dispenses data out of its channel and a source channel end accepts data into its channel.

A connector is a set of channels and channel ends organized in a graph of nodes and edges. Channels are joined together in a node, so, a node is a construct which consists of a set of channel ends.

Reo provide two types of operations: topological –ones that allow manipulation of connector topology and IO – ones that allow input/output of data. Reo enables components to connect and perform I/O on the connector, namely read, take and write. Topological operations are *join* and *split*, because of space limitation we do not explain them here.

As we mentioned earlier, Reo has operational semantics (Arbab, 2003). The semantics of a Reo connector is defined by the composition of the semantics of its channels and nodes. Users define the semantics of channels and Reo defines the semantics of nodes.

Arbab has defined a set of complete channel types (Arbab, 2004), namely *Sync*, *Filter*, *SyncDrain*, *LossySync*, and *FIFO-1*. Figure 1 shows the visual notation for these channels.



Figure 1: Visual notation for basic channels.

The “Exclusive Router 2” connector is shown in Fig 2. This connector has one input and two output ports. When a data is written on the input port, it can be read only by one of the components reading from the output ports.

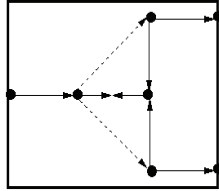


Figure 2: Exclusive Router 2 connector.

We can use abstraction notation to hide the internal structure of the connectors using a box and interface ports on its border. The abstract notation of “Initially Closed Valve” connector is shown in Figure 3. This connector is initially closed which implies that when a data is written on its input port ‘a’, it won’t flow through the connector until a data is written on the ‘c’ port (by the administrator) and the valve is opened.

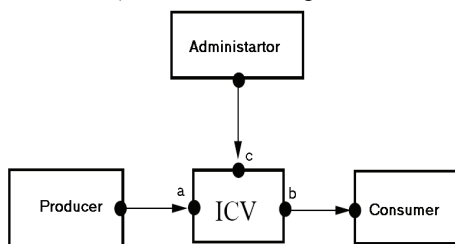


Figure 3: Abstract notation of “Initially Closed Valve” connector.

The “Initially Opened Valve” connector is the same as “Initially Closed Valve” but data can flow from producer to consumer until administrator closes the valve.

4 FORMAL MODEL

As described in Section 1, an organization coordinates agent interactions. Thus an organization can be viewed as a coordination artifact that coordinates the behaviour and interactions of agents in terms of long-term goals of system. We consider organization as an open system, where agents are self-interested and can enter and leave organization dynamically.

A formal model has been proposed in (Omicini et al., 2004) to specify an environment-based coordination artifact. In the environment-based coordination model, agents are coordinated via data existed in the environment. Thus, agents do actions using operations defined by the user interface of the artifact. When artifact receives an action, it is responsible to execute the action, and reify proper data to keep track of agent actions.

These data define the coordination status of the coordination artifact. In this model, agents do not have a direct communication and they communicate via data reserved in the environment. The operating instructions of the artifact define for each agent how to exploit coordination service.

We have extended the formal model proposed in (Omicini et al., 2004) with organizational concepts to specify an organization. The usage interface defines what actions an agent can do and the set of operating instructions defines the interaction protocol between agent and organization. The coordination behaviour of the artifact defines how the organization coordinates the interactions of agents. In our model, agents can communicate directly, which may be synchronous or asynchronous.

An organization is specified by a tuple $\langle R, A, \psi, \alpha, \beta, \rho, \delta, \rightarrow_{\sigma}, \gamma \rangle$. Some of these parameters are in common with the model in (Omicini et al., 2004). The set R defines the set of roles required within the organization to reach its goals. The set A defines the set of agents and the roles they play.

Agent-oriented methodologies such as Gaia (Wooldridge et al., 2000) and Tropos (Giorgini et al., 2004), specify an organization in terms of roles and their interaction relation/structure, which are usually modelled as interaction protocols. In (Grossi et al., 2005) three relations are distinguished between roles, i.e., power, control, and coordination. So, agents can interact by delegating tasks to each other, passing information to each other, or taking responsibility for each other. The meta-variable ψ is the set of binary relations, which defines the organizational structure of MAS in three dimensions of control, power and information:

$$\psi := \{power(r,s), control(r,s), inform(r,s), r,s \in R\}$$

These relations are not limited and users can define other (social) relations. For instance, the power relation specifies the agent enacting role r delegates tasks to the agent enacting role s . Note ψ is exploited to cross-check the consistency between organizational structure, agent interaction protocols and the coordination behaviour. For example when a power relation exists between two roles, the delegating agent is allowed to delegate a task and the delegated agent should receive the task either synchronously or asynchronously.

The meta-variable α ranges over the operations allowed by the organization to the agents and it defines the actions an agent can do/initiate. The meta-variable β ranges over the perceptions of action completion and it may contain some information about the outcome of the action. Therefore, the set L of interactions between agents and the organization, ranged over by l , is defined by the syntax as follows:

$$l ::= id!a \mid id?\beta$$

The $id!a$ represents an agent identifier id , executes an action a , and $id?\beta$ represents agent id perceives the completion β for the action a .

The function ρ associates to each agent identifier id the usage instruction I he is committed to follow in the organization and it defines the admissible actions and perceptions. Instructions can be defined by exploiting typical process algebra operators, i.e. by the syntax:

$$I ::= 0 \mid !a \mid ?\beta \mid I+I \mid I;I \mid I||I$$

Where, 0 is the void instruction, $!a$ is execution of an action, $?\beta$ is perception of a completion, operator “+” is used for choice between instructions, “;” for sequential composition of instructions and “||” for parallel composition of instructions. The definition can be recursive. As an example, the definition $I := !a; (?\beta || I)$ means that the agent is initially allowed to do an action a and later, while it can do the whole protocol again, doing another action of a , it can perceive the completion of previous actions (i.e. β) of a .

The meta-variable δ ranges over the data reified into the organization (like databases or temporary containers) to possibly keep information of organization. Agents may not communicate directly with each other to coordinate their actions, thus they reify data into the organization which then taken by another agent to coordinate their behaviours. The meta-variable σ ranges over the set of Σ of states of the organization, which is defined as follows:

$$\sigma ::= 0 \mid \delta \mid I \mid (\sigma || \sigma)$$

The operator $||$ is characterized by the following rules:

$$\sigma || 0 \hat{=} \sigma, \sigma || \sigma' \hat{=} \sigma' || \sigma, \sigma || (\sigma' || \sigma'') \hat{=} (\sigma || \sigma') || \sigma''$$

Thus, each state σ is defined by the parallel composition of elements δ and interactions l . The l is used to represent the pending actions to be executed and pending completions waiting to be perceived.

The state of organization is changed when an interaction occurs and is modelled by the transition relation $\rightarrow_{\sigma} \subseteq \Sigma \times \Sigma$, representing the fact that a state σ may eventually move to another σ' , when a new pending action has to be computed which typically causes a change in the data reserved into the organization.

The meta-variable γ ranges over the first order predicates to define the organizational rules using propositional logic. It is defined by the syntax as follows:

$$\gamma ::= a \mid \neg\gamma \mid \gamma \wedge \gamma \mid \gamma \vee \gamma$$

Where “ a ” is the set of atomic propositions existed in the organization. These rules usually define the

pre-conditions required for the interactions between agents, or an agent and the organization.

The coordination behaviour of organization is described by a transition system $\langle C, \rightarrow, LY\{\tau\} \rangle$. C is the set of configurations of the organization, which is defined by the composition of ρ and σ shown by $\rho \otimes \sigma$, where the function ρ associate to each agent the instruction it currently has to follow, and the σ defines the current state the organization. The transition function $\rightarrow_{\zeta} C \xi L \xi C$ is defined by below rules:

$$\frac{\rho(id) \xrightarrow{!a} I}{\rho \otimes \sigma \xrightarrow{id!a} \rho[id \mapsto I] \otimes \sigma || id!a} \quad \text{Rule 1}$$

$$\frac{\rho(id) \xrightarrow{?\beta} I}{\rho \otimes \sigma || id ? \beta \xrightarrow{id?\beta} \rho[id \mapsto I] \otimes \sigma} \quad \text{Rule 2}$$

$$\frac{\sigma \xrightarrow{\tau} \sigma'}{\rho \otimes \sigma \xrightarrow{\tau} \rho \otimes \sigma'} \quad \text{Rule 3}$$

The first rule defines an agent id can do/initiate an action, if the $\rho(id)$ allows this action and then this action will reified in the state σ . The second rule defines the completion β to action a ; if this is reified into the state σ and the $\rho(id)$ allows perception of the completion. The third rule is derived from the actual coordination task inside the organization; when the \rightarrow_{σ} defines changes in the states of the organization, there is a silent change in the system configurations, which is shown by a silent transition (τ).

5 MAPPING OF THE FORMAL MODEL TO REO

Organization is a coordination artifact that can be implemented using Reo, an exogenous coordination language as explained in Section 3. We apply the formal model of organization explained in Section 4 to make the implementation of organization by Reo circuits systematically. In this section, we show how to implement an organization given the tuple $\langle R, A, \psi, \alpha, \beta, \rho, \delta, \rightarrow_{\sigma}, \gamma \rangle$.

The set of agent operations within an organization is restricted to the operations that are allowed to the agent by Reo on the connector interfaces: write, read and take. Thus α is a proper subset of I/O operations allowed by Reo for an organization. In Reo, an operation is not started unless it can be completed, so for these completions the β is trivial. These perceptions that contain information should be defined by an extra *write* and *read* operations that will be included in β .

An interaction connector is implemented for the interaction protocol of each role and organization connector is defined according to the transition relation. The organizational rules are implemented by control

connector. When an agent enters into an organization, it is committed to an interaction protocol, which can be implemented by an interaction connector. Agents initiate actions (read, take or write) via Reo circuits which accepts an action if it is admissible, otherwise it cannot be initiated. For the different interaction protocol patterns commonly used (Zlatev et al., 2004), we describe their corresponding Reo circuits. The Reo circuit for the interaction protocol $I:=(a;b);I$ is shown in Figure 4. We call this circuit *Sequencer*. The Reo circuit for the interaction protocol $I:=(a+b);I$ is shown in Figure 5. We call it *Choicer* during the paper.

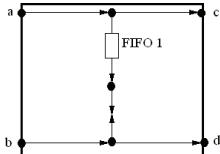


Figure 4: The Reo circuit for $I:=(a;b);I$ protocol.

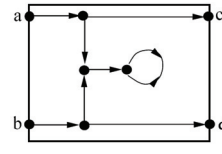


Figure 5: The Reo circuit for $I:=(a+b);I$ protocol.

If an ‘‘Initially Open Valve’’ circuit placed on the way of the output c , the protocol changes into the $I:=(a;b)$ protocol, which is shown in Figure 6.

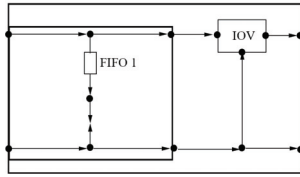


Figure 6: The Reo circuit implements the $I:=(a;b)$ protocol.

The *Choicer* circuit has two parts, namely competitors and choicer parts as shown in Figure 7.

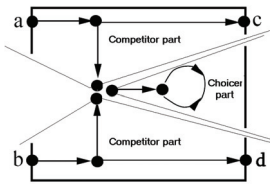


Figure 7: Different parts of the Choicer circuit.

The Reo connector implementing parallel composition is defined in (Ghassemi, 2006). The \rightarrow_σ can be implemented by *Sync*, *SyncDrain* and *FIFO* channels. The *FIFO* channel has the ability to store data. Thus we can use different capacities of *FIFO* to store data reified into the organization. The δ meta-variables define the outputs of agents that should be stored in a *FIFO*

channel. The \rightarrow_σ defines when a *FIFO* gets full and empty or which actions should be synchronized.

The meta-variable γ place some restrictions between an agent interaction and the organization or between agent interactions. They can be implemented by control connectors, which are placed between agents and interaction connectors or within organization connector and control the flow of data in the connectors. Control connectors are usually implemented by ‘‘Initially Open Valve’’, ‘‘Initially Closed Valve’’ and *SyncDrain* channel (Ghassemi, 2006).

Reo is a compositional model and complex circuits are composed of simpler ones, which enable us to define interaction protocols compositionally. We can define complex interaction protocols by composing simpler ones. We define three functions over a protocol namely *first*, *last* and *tail*. The *first* function returns actions that initiate the protocol. The *tail* function returns the protocol by omitting the first actions from the protocols. The definition of *first* function is shown as follows:

$$first(I) = \begin{cases} a & a \in \alpha \cup \beta \\ \bigcup_{I_i \neq I} first(I_i) & (I_1 + I_2 + \dots + I_n) \vee (I_1 || I_2 || \dots || I_n) \\ first(I_1) & (I_1; I_2; \dots; I_n) \end{cases}$$

The *last* function defines the actions should be done in order a protocol get finished. The definition of *last* function is shown as follows:

$$last(I) = \begin{cases} a & a \in \alpha \cup \beta \\ \bigcup_{I_i \neq I} last(tail(I_i)) & (I_1 + I_2 + \dots + I_n) \vee (I_1 || I_2 || \dots || I_n) \\ last(tail(I_1)) & (I_1; I_2; \dots; I_n) \end{cases}$$

To define the composition of protocols under an operator we specify the $Compose(I_1, I_2, \dots, I_n, op)$ function for different *op* operations. Thus the Reo circuit for $Compose(I_1, I_2, \dots, I_n, +)$ is implemented by implementing the Reo circuit of $Compose(first(I_1), first(I_2), \dots, first(I_n), +)$ where $Compose(a, b, \dots, +)$ is implemented using *Choicer-n* circuit. This circuit is easily implemented by connecting competitor parts to the choicer part of a *Choicer* circuit. The Reo circuit for $Compose(I_1, I_2, \dots, I_n, ;)$ is implemented by implementing the $Compose(last(I_1), first(I_2), ;)$ to $Compose(last(I_{n-1}), first(I_n), ;)$, where $Compose(a, b, ;)$ is implemented by *Sequencer* circuit.

The transition relation defines how the agent actions should be coordinated with each other. In this step, designer can decide to coordinate two actions synchronously or asynchronously. For example when a manager delegates a task to his employer, the send and receive actions of manager and employer should be

coordinated. If they are synchronous, the appropriate ports of their interaction protocols are connected directly to each other; otherwise, a *FIFO-1* channel is used to keep the delegation, which is later taken by the employer.

Each agent that enacts a set of roles needs a set of Reo circuits implementing the interaction protocols required for each role.

6 EXAMPLE

In this section we specify and implement an online store, where seller and buyer interact with each other. The buyer can ask about the price of items and pay money for an item and get the item. In return the seller answers client questions and receives money. The seller should opens the store before any client can enter and does any interaction.

6.1 Formal Specification

The online store is an organization where the two roles, seller and buyer exist (R). Thus agents playing role within the store are seller and buyer each have roles of seller and buyer respectively (A). There is a flow of the information from the seller to the buyer in order to inform the buyer about the price of items and there is also a flow of money from the buyer to seller. Thus the organizational structure is defined as follows:

$$\psi ::= \{ \text{inform}(\text{seller}, \text{buyer}), \text{pay}(\text{buyer}, \text{seller}) \}$$

The actions that the buyer and the seller agents can do in the organization are shown as follows:

$$\begin{aligned} \alpha & ::= \text{answer} \mid \text{ask}(\text{item}) \mid \text{get_money} \mid \text{pay_money}(\text{amount}) \mid \text{open} \\ \beta & ::= \text{OK_answer} \mid \text{OK_ask}(\text{price}) \mid \text{receive_money}(\text{amount}) \mid \text{OK_pay}(\text{item}) \mid \text{OK_open} \end{aligned}$$

The interaction protocols (ρ) for each agent is defined according to the organizational structure as follows:

$$\begin{aligned} \text{Seller} & ::= (!\text{answer}; ?\text{OK_answer}) + (!\text{get_money}; ?\text{receive_money}(\text{a})) + \text{open}; \text{Seller} \\ \text{Buyer} & ::= ((\text{ask}(\text{item}); ?\text{OK_ask}(\text{price})) + (\text{pay_money}(\text{a}); ?\text{OK_pay}(\text{item}))) ; \text{Buyer} \end{aligned}$$

The δ contains “start” which indicates that the seller has opened the store. The transition relation \rightarrow_σ is defined by the two simple rules:

$$\begin{aligned} \text{id}_{\text{seller}}! \text{open} & \rightarrow \text{Start} \parallel \text{id}_{\text{Seller}}? \text{open} \\ \text{Start} \parallel \text{id}_{\text{seller}}! \text{answer}(\text{item}) \parallel \text{id}_{\text{buyer}}! \text{ask}(\text{item}) & \rightarrow \text{Start} \parallel \\ \text{id}_{\text{seller}}? \text{OK_answer} \parallel \text{id}_{\text{buyer}}? \text{OK_ask}(\text{a}) & \\ \text{Start} \parallel \text{id}_{\text{seller}}! \text{get_money} \parallel \text{id}_{\text{buyer}}! \text{pay_money}(\text{a}) & \rightarrow \text{Start} \parallel \\ \text{id}_{\text{seller}}? \text{receive_money}(\text{a}) \parallel \text{id}_{\text{buyer}}? \text{OK_pay}(\text{item}) & \end{aligned}$$

We can consider an organizational rule that when a buyer can ask or pay money if the seller opens the store. This rule is defined in the following:

$$\text{Start } f(\text{id}_{\text{Buyer}}! \text{ask} \oplus \text{id}_{\text{Buyer}}! \text{pay})$$

In this formal specification, we can check the correctness of interaction protocols and the coordination behaviour of the organization according to the organizational structure.

6.2 Reo Implementation

In this section we specify and implement online store according to the formal specification and mapping rules explained in Section 5. The α and β are redefined as follows:

$$\begin{aligned} \alpha & ::= \text{read}_{\text{answer}}(\text{item}) \mid \text{write}_{\text{ask}}(\text{item}) \mid \text{read}_{\text{get_money}}(\text{amount}) \mid \\ & \text{write}_{\text{pay}}(\text{amount}) \\ \beta & ::= \text{read}_{\text{ask}}(\text{price}) \mid \text{read}_{\text{pay}}(\text{item}) \end{aligned}$$

The interaction protocol for agents is redefined as follows:

$$\begin{aligned} \text{Seller} & ::= ((\text{read}_{\text{get_qu}}(\text{item}); \text{write}_{\text{answer}}(\text{price})) + (\text{read}_{\text{get_money}}(\text{amount}); \text{write}_{\text{give_item}}(\text{item})) + \text{write}_{\text{open}}); \text{Seller} \\ \text{Buyer} & ::= ((\text{write}_{\text{ask}}(\text{item}); \text{read}_{\text{answer}}(\text{price})) + (\text{write}_{\text{pay}}(\text{amount}); \text{read}_{\text{get_item}}(\text{item}))); \text{Buyer} \end{aligned}$$

The interaction connectors for each agent are defined according to the interaction protocols and the interaction connectors are connected to each other according to the transition relation (organization connector). The implementation of the store is shown in Figure 8.

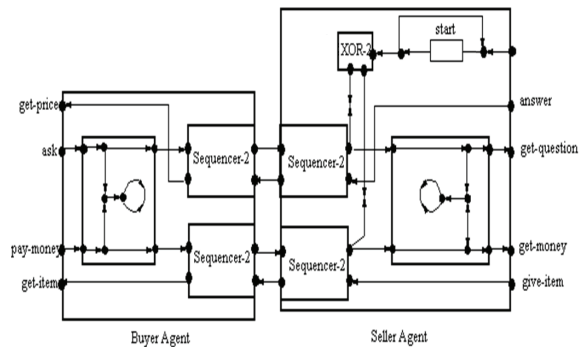


Figure 8: Implementation of online store by Reo.

7 CONCLUSIONS

In this paper, we propose a formal model for the specification of multi-agent organizations. This specification formally defines what tasks an agent is allowed to do in an organization and in synchronization with which actions in the system. It also defines what the pre-condition of each task is and how the organizational structure affects on the interactions between agent and the coordination behaviour of organization.

We apply Reo coordination language to implement organizations. To make the implementation systematically, we use our formal model to specify the system according to the Reo I/O operations. We define how to convert specification to the implementation by introducing some Reo circuit for common interaction protocols and how they can be composed to make complex protocols. Reo not only provides a formal specification but also provide an implementation. Thus the Reo circuits are executable. There is a tool to run Reo circuits (Dave, 2005).

We are going to find a mapping between process algebra expressions into the Reo circuit. This mapping enables us to automate the conversion of specification by our formal model to a Reo circuit.

REFERENCES

- Arbab, F., 2003. Abstract behavior types: A foundation model for computers and their composition. In *Proceeding of the First International Symposium on Formal Methods for Components and Objects (FMCO 2002)*, vol. 2852, pp 33–70.
- Arbab, F., 2004. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14:329 – 366.
- Arbab, F. and Rutten, J., 2003. A coinductive calculus of component connectors., *Recent Trends in Algebraic Development Techniques, Proceedings of 16th International Workshop on Algebraic Development Techniques (WADT 2002)*, volume 2755, pages 35–56.
- Clarck, D., 2005. Reo Lite. Internal report, Software Engineering Department, CWI.
- Dastani, M., Arbab, F., and de Boer, F. S., 2005. Coordination and composition in multi-agent systems. In *4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 2005, Utrecht, Netherlands*.
- DeLoach, S., 2002. Analysis and design of multi-agent systems using hybrid coordination media. In *proceedings of Software Engineering in Multiagent Systems*.
- DeLoach, S. and Matson, E., 2004. An organization model for designing adaptive multiagent systems. In *The AAAI-04 Workshop on Agent Organizations: Theory and practice*.
- Ghassemi, F., 2006. Analysis and Design of Multi-agent Systems using Reo. Master Thesis, Isfahan University.
- Giorgini, P., Kolp, M., Mylopoulos, J., and Pistore, M., 2004. The Tropos Methodology: An Overview. In *Methodologies and Software Engineering for Agent Systems, Kluwer*.
- Grossi, D., Dignum, F., Dastani, M., and Royakkers, L. M. M., 2005. Foundations of organizational structures in multiagent systems. In *4rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pp 690–697.
- Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., and Tummolini, L., 2004. Coordination artifacts: Environment-based coordination for intelligent agents. In *proceeding of 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 286–293.
- Wooldridge, M. and Jennings, N., 2000. The Gaia Methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.
- Zambonelli, F., Jennings, N. R., and Wooldridge, M., 2000. Organizational abstractions for the analysis and design of multi-agent systems. In *Agent-Oriented Software Engineering, First International Workshop*, pages 235–251.
- Zlatev, Z., Diakov, N., and Pokraev, S., 2004. Construction of negotiation protocols for e-commerce applications. *ACM SIGecom Exchanges*, 5(2):12-22.