Available online at www.sciencedirect.com

SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science 159 (2006) 241–264

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

From Theoretical e-barter Models to an Implementation Based on Web Services [1]

Mario Bravetti, Adalberto Casalboni

Università di Bologna, Dipartimento di Scienze dell'Informazione, Mura Anteo Zamboni 7, 40127 Bologna, Italy, E-mail: bravetti@cs.unibo.it

Manuel Núñez, Ismael Rodríguez

Dept. Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain, E-mail:{mn,isrodrig}@sip.ucm.es

Abstract

An e-barter system is an e-commerce environment where transactions do not necessarily involve money. They are multi-agent systems where agents perform exchanges of resources on behalf of their respective users. Besides, their structure is based on a tree of markets. In this paper we develop a suitable *design* for this kind of systems. The design will be defined by means of a *web services* by using BPEL4WS. Since the formal specification abstracts most practical details, the development of such design definition requires to face several challenges.

Keywords: e-commerce, e-barter, multi-agent systems, web services

1 Introduction

Among those areas where the development of Computer Science has changed our society during the last years, the relevance of e-commerce technologies is remarkable. New mechanisms to perform transactions have appeared and they entail new challenges. Since e-commerce systems dramatically affect user's

[1] Research partially funded by EU Integrated Project Sensoria, contract n. 016004, by the Spanish *Ministerio de Ciencia y Tecnología* project MASTER (TIC2003-07848-C02-01), the *Junta de Comunidades de Castilla-La Mancha* project DISMEF (PAC-03-001), the Marie Curie Research and Training Network *TAROT* (MRTN-CT-2003-505121).

doi:10.1016/j.entcs.2005.09.034

possessions, their reliability is specially important for their success. Similarly to other domains, formal specifications allow to predict relevant properties of e-commerce systems and they provide a reference of ideal behavior for developing an implementation. In order to provide handleable and unabiguous models, formal languages abstract from some low level details that are considered irrelevant for the system description. In spite of the gap between specification models and final implementations, specifications provide a developer with a model of ideal behavior of the system that may be useful not only in the *analysis* phase of a project but also during the whole development process.

Among existing applications of formal methods to the area of e-commerce (see e.g. [12,4,11,2,10]), here we consider the formal definition of e-barter systems formerly introduced in [6] and extended in [7,9]. This kind of systems are characterized because transactions do not necessarily involve money. They provide a generalization of the classical e-commerce model because they can represent systems where exchanges may or may not involve the use of money. Users connected to the system exchange goods according to their respective preferences. Exchanges are performed until the system reaches a kind of optimal distribution of resources with respect to the user requests. In order to avoid critical bottlenecks in the system performance, a single global market is replaced by a hierarchical tree-like structure of markets, where parallelism can be exploited allowing local markets to progress independently to each other. Following this structure, users are connected to different local markets in the leaves of the tree according to proximity reasons. Exchanges are performed inside each local market until it reaches an optimal distribution. Then, the market *becomes* an agent that acts as representative of the users that traded inside it. This agent exchanges resources on behalf of its users with other (representative) agents in a *higher level* market until this higher order market reaches an optimum. The mechanism is repeated until the global market, that is the one at the top of the hierarchy, reaches an optimum. The behavior of this system is formally defined in terms of a process algebraic notation. This definition allows to formally reason about the behavior of the system and predict some relevant theoretical results. In particular, as it is shown in [9], if an adequate amount of information is exchanged among the different levels, the final distribution of resources is *optimal* from the point of view of the users. That is, the hierarchical system reaches the same kind of distribution that can be reached by a non-hierarchical system where a single central market would embrace all the agents (i.e. the *economic* efficiency of a hierarchical market matches that of a non-hierarchical market).

In this paper we face the problem of developing a design of the e-barter system which may constitute the basis for an efficient implementation. *Web* *services* and the related notion of orchestration of services constitute a suitable *conceptualization semi-fomal model* allowing to move from an abstract model to an implementable distributed model that allows us to exploit parallelism of the specified system. Web Services related technologies are a set of middleware technologies for supporting *Service Oriented Computing* [5]. One of the main goals of this paradigm consists in enabling the construction of a network of integrated collaborative applications regardless of the platform and the development language. Web services technologies allow to construct these systems by supporting their definition in a structured and standardized way. In particular, a web service is identified by a URI, and public interfaces and bindings are defined by using XML so that their definition can be discovered by other software systems [15]. Web services allow to publish services on the internet by means of a naming service called UDDI [14] (*Universal Description, Discovery and Integration*). XML based WSDL technology [3] (*Web Services Description Language*) is used to describe services inside UDDI naming servers and making them accessible at a certain location by means of an XML based protocol called SOAP [13] (*Simple Object Access Protocol*). Orchestration of web services is supported by BPEL4WS [1] (*Business Processes for Web Services*), which is a language for describing web-service behavior (workflow) in terms of calls to other web-services. It can be used for the design and implementation (since it is really executable by means of engines) of systems that are defined in terms of composition of web-services.

The representation of the e-barter system as orchestration of web services requires to address several practical challenges and force the developer to define more in the detail the system architecture and behavior: the entities involved (which as we will see can be both concrete, i.e. real services on the internet, or abstract, i.e. just representing activities made by human beings), the flow of interaction among these entities, the kinds of data exchanged, the time for message exchange, timeout for receiving requests, etc. Starting from the system formal specification we will develop a design of a possible implementation, where entities and flow interactions are defined in such a way as to obtain an efficient implementation which exploits system parallelism of different local markets expressed as distributed services. We will also note that an alternative design is possible which increases the performance at the price of loosing optimality of the solution in certain cases.

The rest of the paper is structured as follows. In Section 2 we sketch the formal model defining e-barter systems. In Section 3 we present the main design decisions for implementing e-barter systems via web-services. In Section 4 we present the main diagrams for the BPEL4WS processes describing the involved entities. Finally, in Section 5 we present our conclusions. The appendix contains the operational semantics of the formal model terms.

2 Formal specification of multi-level e-barter systems

In this section we briefly describe e-barter systems and introduce their formal specification. A summary of the formal operational semantics denoting the behavior of these systems can be found in the appendix of this paper. An e-barter system is a multi-agent system where agents exchange resources on behalf of their respective users. Since agents must perform exchanges according to the preferences of users, we need a suitable notation to denote preferences. We use *utility functions.* The input of a utility function is a *basket* of resources and the output is a *numerical value* denoting the preference on this basket. Let f_A be the utility function of an agent A. Let $\bar{x}$ be a basket with 2 apples and 1 euro, and $\bar{y}$ be another basket with 1 apple and 2 euros. If $f_A(\bar{x}) > f_A(\bar{y})$ then A *prefers* $\bar{x}$ to $\bar{y}$. A possible utility function showing that behavior is $f_A(apples, euros) = 2 \cdot apples + euros$. Let us suppose there is another agent B whose utility function is $f_B(apples, euros) = 2 \cdot apples + 3 \cdot euros$. Then, if the agents A and B perform an exchange where agent A gives 1 euro to B and B gives 1 apple to A, then both utility functions return higher values after the exchange, that is, both agents *improve.*

An e-barter system performs *fair* exchanges, that is, exchanges where at least one agent improves and none of them worsens. When no more fair exchanges are available, the market is *completed*, that is, it reaches a configuration that cannot be improved. Instead of using a single market where all agents in the system exchange resources, agents will be grouped in local markets according to proximity reasons. For example, agents belonging to the same city are put together until their markets are completed. Then, they use some *representatives* to exchange resources in a higher order market involving several cities. After these are completed, new representatives exchange resources in a higher market, and so on.

Initially, customers willing to participate in an e-barter system are *represented* by (electronic) agents. These agents are provided with two parameters: The *basket of resources* that the customer is willing to exchange and a *utility function.* Such agents trade in the most local market whose area includes the location of the the related customers.

Then the e-barter system works according to the following algorithm:

(1) Agents exchange goods inside their local market. A multilateral exchange will be made if (at least) one of the involved agents improves its utility and none of them decreases its utility. This is repeated until no more exchanges are possible. In this case we say that the local market is

completed (or *saturated*).

(2) Once a market is completed, their agents are combined to create a new agent which will trade in the higher level market whose area include the location of the current market (this happens unless we are in the top-level market). This agent behaves as a representative of the combined agents. The new agent will have as basket of resources the union of the baskets corresponding to each agent. Its utility function will encode the utilities of the combined agents. This utility function imposes that resources obtained by the representative are such that they can delivered among its users in such a way that no user worsens with respect to the previous distribution. If this condition holds, it returns the addition of utilities of each represented user (see [9] for more details). *First order* agents will be combined again into markets, according to *proximity* reasons.

(3) The exchanges in the higher level market are performed starting again from (1), unless we are in the top-level market. In this case the process is finished and the distribution of goods obtained is the final one.

Note that during the whole process, after the completion of any market, the formal model keeps track of distribution of goods to the several customers by propagating such information in a top-down way through the tree of markets until it arrives to the leaves of the tree (i.e. to the agents directly representing customers).

As we will see in Section 3, the design of the e-barter system will address several practical issues that are beyond the detail level considered in the previous description. Some of these issues will make us to *reconsider* the previous scheme. For example, requiring that all subagents are connected to a market before it becomes completed is actually needed to provide final optimal distributions, but this requirement might not be feasible in practice. For instance, if an agent is temporally out, then it may block the rest of agents. So, when a market becomes a new higher order market (see step (2)), in some situations it will be able to become a representative of only *some* agents. These changes will affect the operational semantics of the formal language (presented in the appendix). Note that these changes show that the relation between the formal specification (i.e., the model of the analysis of the system) and the web services definition we construct from it (i.e., the design and implementation) is two-fold.

2.1 A brief introduction to the formal model

Next we briefly introduce the formal representation of e-barter systems. The formal definition of an e-barter system is made by means of a specification language that was explicitly developed to define this kind of systems. An e-barter system is formally given by a syntactical term in the syntax of this language. The semantics of the language implicitly define the behavior of an e-barter system in a formal fashion. A brief introduction to the operational semantics can be found in the appendix of the paper. Even though this language uses a process algebraic notation (mainly when defining the operational rules) it does not need the usual operators appearing in this kind of languages (choice, restriction, etc). In fact, our constructions remind a parallel operator as the one presented, for example, in the process algebra CCS [8].

Definition 2.1 A *market system* is given by the following EBNF:

$$\begin{aligned}
MS &::= ms(M) \\
M &::= A \mid \mathtt{unsat}(M, \ldots, M) \\
A &::= (S, u, \overline{x}) \\
S &::= [\,] \mid [A, \ldots, A]
\end{aligned}$$

□

First, in order to avoid ambiguity of the grammar, we annotate market systems with the terminal symbol ms. Intuitively, the market $M = (S, u, \overline{x})$ (that is, $M = A$) represents a *completed* market, that is, a market where no more exchanges can be performed among its agents. Let us note that in this case the market represents an *agent* that will be able to make transactions with other agents in a higher market. In the previous expression, u denotes the utility function of M and $\overline{x}$ represents the basket of resources owned by M. We consider that there are p different commodities,[2] that is $\overline{x} \in \mathbb{R}^p_+$, and that the amount of *money* is placed in the last component of the tuple.

Regarding the first argument of M there are two possible situations. Either S is an empty list or not. In the first case we have that M represents an *original* agent, that is, a direct representative of a customer (note that a single agent is trivially completed since there is nobody to deal with). In the second case, if $S = [A_1, \ldots, A_n]$ then we have that M represents an agent associated with the (possible higher order) agents $A_1, \ldots, A_n$ belonging to a completed market.

The second possible syntactic form of M, $\mathtt{unsat}(M_1, \ldots, M_n)$, represents an *uncompleted* market consisting of the markets $M_1, \ldots, M_n$. Let us remark that in this case some of the sub-markets may be completed.

[2] We are assuming that all the items are *goods*. Nevertheless, agents could also trade *bads*. For example, a customer would be willing to give an apple pie if he *receives* minus s brown leaves in his garden. However, bads are usually not considered in microeconomic theory, as they can be easily turned into goods: Instead of considering the amount of leaves, one may consider the absence of them.

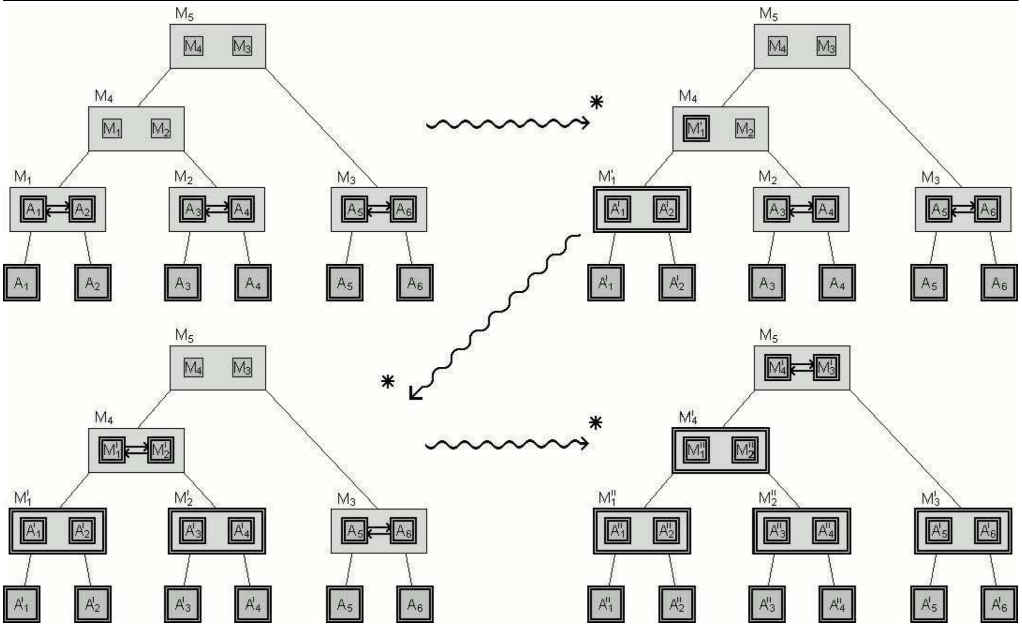


Fig. 1. A market system and some operational transitions.

Next we present an example showing how an e-barter system may be constructed. In this example we will also (informally) introduce the operational transitions of the language.

Example 2.2 Let us consider a system with six agents $A_i = ([\ ], u_i, \overline{x_i})$, for $1 \leq i \leq 6$. We suppose that these agents are grouped into three different markets. Initially, these markets are uncompleted (uncompleted markets are represented by a single square in the figure), so we make the following definitions:

$$M_1 = \texttt{unsat}(A_1, A_2)$$

$$M_2 = \texttt{unsat}(A_3, A_4)$$

$$M_3 = \texttt{unsat}(A_5, A_6)$$

Let us consider that the first two markets are linked, and the resulting market is also linked with the remaining market M_3. We should add the following definitions:

$$M_4 = \texttt{unsat}(M_1, M_2)$$

$$M_5 = \texttt{unsat}(M_4, M_3)$$

Finally, the global market is defined as $M = ms(M_5)$. This hierarchical structure is graphically presented in Figure 1, top-left.

Following the philosophy explained in the previous section, transactions will be made within a market only among completed sub-markets. So, initially only M_1, M_2, and M_3 are allowed to perform transactions (as we remarked before, original agents are trivially completed).

We will use the symbol $\rightsquigarrow$ to denote exchange of resources. Let us suppose that, after some exchanges, M_1 gets completed. That is, there exists a sequence of exchanges $M_1 \rightsquigarrow M_1^1 \rightsquigarrow M_1^2 \cdots \rightsquigarrow M_1^n = M_1'$ such that $M_1' \not\rightsquigarrow$. In this case, the market grouping the first two agents should be labeled as completed. So, the agents effectively perform all the achieved transactions becoming A_1' and A_2', respectively. Then, the first market will be turned into $([A_1', A_2'], f(u_1, u_2), \overline{x_1} + \overline{x_2})$, where f is a function combining utility functions which works as described in the algorithm presented in the previous section (step 2). In parallel, M_2 will have a similar behavior.

Once both M_1 and M_2 get completed, transactions between them will be allowed. Note that these transactions (inside the market M_4) will be performed according to the new utility functions, $f(u_1, u_2)$ and $f(u_3, u_4)$, and to the new baskets of resources, $\overline{x_1} + \overline{x_2}$ and $\overline{x_3} + \overline{x_4}$.

The process will iterate until M_5 gets completed. At this point, the whole process finishes. $\square$

Note that if we assume that markets can become completed even if all the submarkets are not completed yet and that new higher order agents can represent only a subset of agents, then other sequences of interaction may occur in the previous example. Despite the presentation of the formal model in [6,7] does not consider these features, the formal model considered in this paper includes them. As we said before, they are added to the system in order to take into account some practical issues that will be addressed in the system design phase.

3 Design of multi-level e-barter Systems via web services

The specification presented in [6,7] and sketched in the previous section presents e-barter systems in terms of what can be done, for example, in terms of constraints on exchanges (they must satisfy the requirements imposed by utility functions). In the design phase, instead, we have to define the precise structure (architecture) of the system, the entities which play a role in it, the order in which things have to be done, and the temporal behaviour of such entities. For example, it has to be taken into account the location of the instances of such entities as well as which instances communicate with each other, the kind of interactions that the entities may perform, the kind of data exchanged, and the workflow of such interactions. When producing the BPEL4WS description of the e-barter system the first thing to do is to identify the different entities (which in the BPEL4WS specification will be denoted by so-called portTypes) that are involved in the system. In particular, not only *concrete entities* (that will be actually implemented via web-services) but also *abstract entities* which just correspond to human actions. In the case of the e-barter system we have three main entities: clients, agents and markets. Each client belongs to a certain market of level 0 (the most local level in the hierarchy of markets). Agents in charge of making exchanges for clients belong to a certain market at level 0, while agents in charge of making exchanges for markets of level $n \geq 0$ belong to a certain market of level $n+1$. It also works the other way around, that is, if a particular market is located at level 0 then it determines a set of clients (we suppose that they are numbered from 1 on) and a set of agents.[3] On the contrary, if the market is located at a level $n > 0$ then it determines a set of agents only (numbered from 1 on) where each agent makes exchanges for a given market of level $n-1$.

In order to represent the architecture of a single market, the operations needed to manage the market have to be effectively provided by the system. What can be seen as a set of operations to manage the data involved in the market becomes an entity to support the behavior and evolution of the market. This entity is called the *manager of the exchange matrix.*

The behavior of the three main entities is defined via orchestration. Their behavior is therefore defined via the semi-formal workflow language BPEL4WS in such a way that is totally consistent with the formal model given in the previous section. We will consider that the formal model includes the additional features described in previous sections, that is, markets can become completed even if some submarkets are not, and new higher order markets may represent only a subset of markets (the operational semantics of the model, presented in the appendix, actually include them).

Conceptually, the design defined in BPEL4WS explicitly represents the communication between the entities, that in the formal model are simply included syntactically one inside each other. More precisely, according to the formal model, a market $\texttt{unsat}(M_1, \ldots, M_n)$ includes syntactically all the agents that will trade in it, i.e. given $I = \{s_1, \ldots, s_r\} \subseteq \{1, \ldots, n\}$ such that $\{M_{s_1}, \ldots, M_{s_r}\}$ is the set of submarkets represented by trading agents, we have that, for all $i \in I$, M_i follows the agent form $(S_i, u_i, \overline{x_i})$. The corresponding market entity in the BPEL4WS design will first communicate with all the agent entities corresponding to $M_{s_1}, \ldots, M_{s_r}$ in order to get their utility functions $u_{s_1}, \ldots, u_{s_r}$ and baskets $\overline{x_{s_1}}, \ldots, \overline{x_{s_r}}$, and then it will perform exchanges according to the internal behavior of $\texttt{unsat}(M_1, \ldots, M_n)$ in the formal model

[3] These agents are numbered as well and we suppose, for the sake of simplicity, that the agent i makes exchanges for client i.

until it saturates. [4] When this happens (supposing that we are not in the top level market), in the formal model we have that $\texttt{unsat}(M_1, \dots, M_n)$ is turned into the agent $A \equiv ([A_1, \dots, A_r], u, \overline{x})$, (which is in charge for trading for this market in the upper level) where u and $\bar{x}$ are the aggregated utility function and the overall basket of resources. In the same way, the market entity will simply transmit the same information to the agent entity (of the higher level) corresponding to A. The behaviour of the agent entity in the BPEL4WS design is simply defined in such a way that it checks the information received and delivers it to its local market for trading (so the described flow repeats).

In addition to the formal model, the BPEL4WS specification considered in the following includes also an explicit phase of transmission of results of the exchange performed to the clients. Such a transmission is started when the top-level market is saturated and it is propagated top-down by markets and agents until the client entities are reached and notified.

Here a different design choice can be considered. In order to improve the efficiency of the system and to simplify the structure of the utility functions obtained by aggregation after the saturation of markets, it is possible, e.g., to decide to propagate upwards only requests of agents who did not perform any exchange in the current market. On the contrary, baskets of agents that performed exchanges can be immediately distributed top-down to the clients. It is easy to see that the price for the improved efficiency is the loss of global optimality of exchanges in general.

In the following section, we describe the mechanism to provide a uniform specification of the markets in our e-barter system by means of local naming and the technique based on UDDI to connect markets to agents of higher level and vice-versa (thus defining the structure of the system: this was done correspondingly by syntactical inclusion of markets in the formal model).

3.1 Local naming definition for each market

In order to invoke a service, other services must know where the first one is located. This mechanism is implemented through the binding of the names of the services to a physical address by using a UDDI server. In the case of our e-barter systems, following a hierarchical architecture, the structure of the UDDI service is depicted in Figure 2. In this graphic we can observe that there exists a different local UDDI server for every market.

Thus, we adopt the idea that each single market of the market system has

[4] The BPEL4WS design abstracts from computation inside atomic services such has the evaluation of the exchanges to be made (the saturation of the matrix of exchanges). Therefore a legal implementation of such services must be made in such a way that it conforms with the formal model behavior.

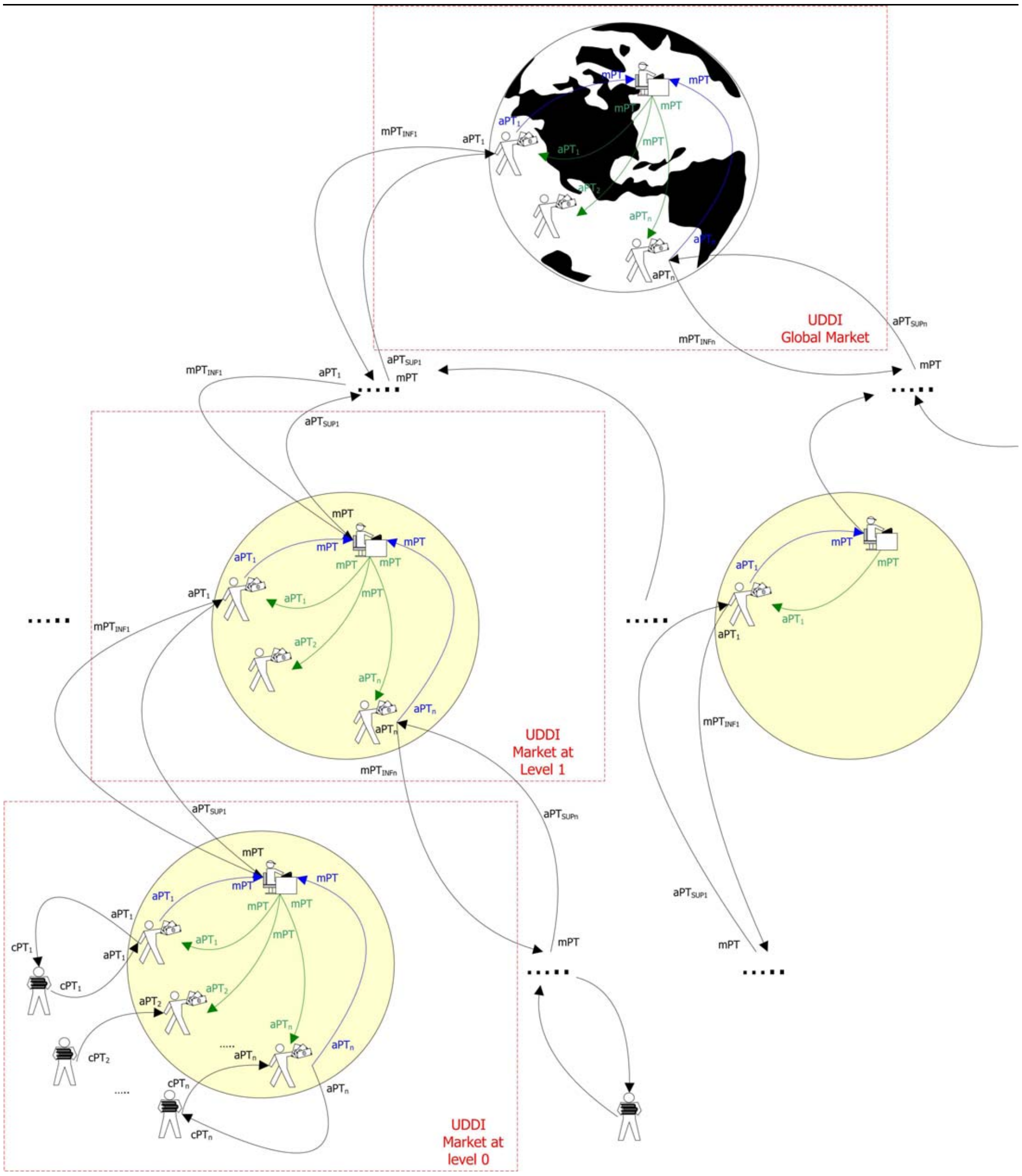
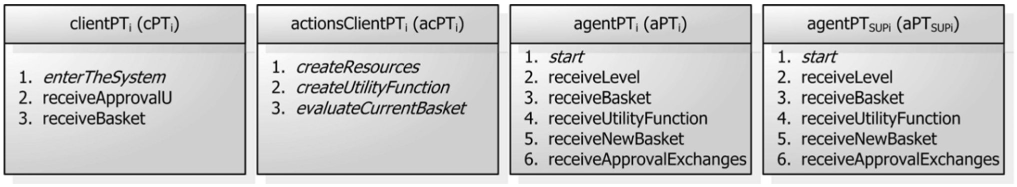


Fig. 2. Scheme to bind names.

its own *namespace*, defined just through a service of a UDDI. In each of these namespaces, every portType comes associated/bound to a specific web service. Thus, if one invokes a service appearing with the same name (same portType) in two different markets then the called service is the one determined by the UDDI associated with the market in which the invocation was performed. In this way, the specific hierarchical structure of e-barter systems can be uniform, that is, the same BPEL4WS process can be used for equal entities (regardless,



clientPT$_i$ (cPT$_i$)	actionsClientPT$_i$ (acPT$_i$)	agentPT$_i$ (aPT$_i$)	agentPT$_{SUPi}$ (aPT$_{SUPi}$)
1. *enterTheSystem*	1. *createResources*	1. *start*	1. *start*
2. receiveApprovalU	2. *createUtilityFunction*	2. receiveLevel	2. receiveLevel
3. receiveBasket	3. *evaluateCurrentBasket*	3. receiveBasket	3. receiveBasket
		4. receiveUtilityFunction	4. receiveUtilityFunction
		5. receiveNewBasket	5. receiveNewBasket
		6. receiveApprovalExchanges	6. receiveApprovalExchanges

Fig. 3. List of abstract portTypes.

e.g., of the level where they are located). The only difference is that the *client* portType is available only in the first level since instances of this type can communicate only with agents located in the first level.

If we are working in a market at a certain level, it is possible to refer to the agent aPT$_i$ of the associated market at the immediately higher level by using the aPT$_{SUPi}$ portType (it is assumed that the UDDI of the market is defined in such a way that aPT$_{SUPi}$ binds to the i-th agent of the market at the higher level). Similarly, given a market at a certain non-zero level, it is possible to refer to the the market mPT at the immediately lower level for which the i-th agent of the market is in charge, by using the mPT$_{INFi}$ portType.

3.2 WSDL definition of local PortType names

In the following, we describe the meaning of portTypes that we use and their operations. In general names of operations are quite self-explaining. [5] Port-Types corresponding to abstract entities are listed in Figure 3, while concrete portTypes are listed in Figure 4.

- clientPT$_i$, agentPT$_i$ and marketPT correspond to the three main entities whose behavior is specified with BPEL4WS in the following.
- agentPT$_{SUPi}$ is a portType identical to agentPT$_i$. It contains the same operations of a the agent agentPT$_i$ and it has the same behavior. The only difference is that it is bound by the UDDI service of a given market to a different location: the location of the agent agentPT$_i$ (in the immediately higher level market) associated to such a market.
- marketPT$_{INFi}$ is, similarly, a portType identical to marketPT. It contains the same operations of a market marketPT and it has the same behavior. The only difference is that it is bound by the UDDI service of a given market to a different location: the location of the market marketPT (in the immediately lower level) associated to the i-th agent of such a market.
- actionsClientPT$_i$ represents the decisions that a customer takes.

[5] Operations in italic are I/O operations, while the other operations are Input only.

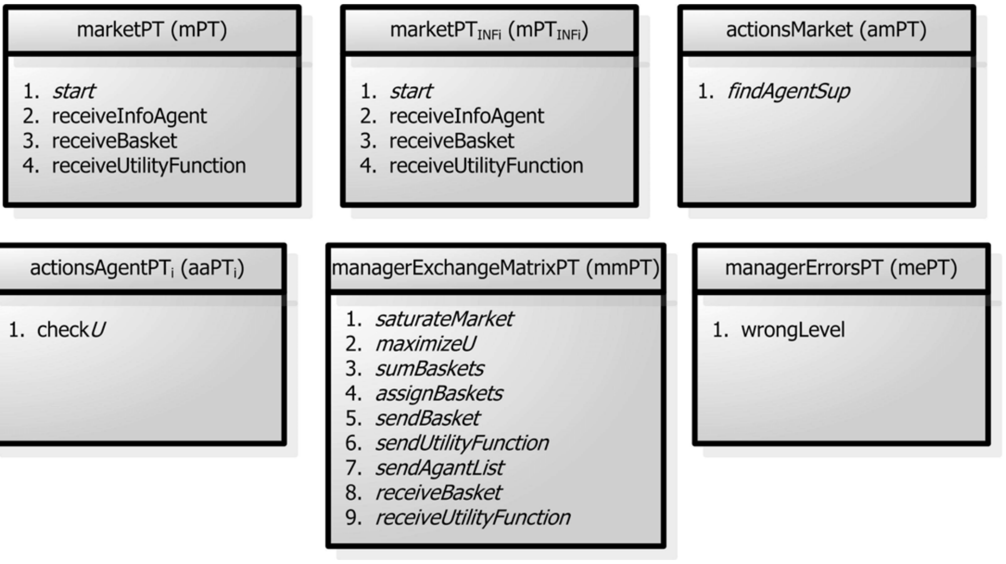


Fig. 4. List of concrete portTypes.

- actionsAgentPT$_i$ is a concrete entity representing actions of agents not directly represented in BPEL4WS. It contains a concrete service for controlling validity of utility functions.
- actionsMarket is a concrete entity representing actions of agents not directly represented in BPEL4WS. It contains a service that returns the index i of the agent in charge of the current market in the higher level. It returns 0 if the current market is the top-level market.
- managerErrorsPT is an entity used for notification of errors.
- managerExchangeMatrixPT is the entity that offers more services to the system. It is used to store the matrix of exchanges for a given market (and related information) and to manage it in an efficient way. It includes operation for the updating, modification and reading of such a matrix. In particular, *saturateMarket* saturate the matrix evaluating an optimal solution in an unspecified way; *maximizeU* and *sumBaskets*, compute the aggregated utility function and the union of baskets after saturation, respectively; *assignBaskets* distribute an updated basket of resources received from the higher level to the agents trading in this market; and *sendAgentList* and *sendBasket* return an array with the list of indexes of agents trading in the current market and return the basket of resources assigned to the specified agent, respectively.

4 BPEL4WS processes for the entities of e-barter systems

In this section we give the BPEL4WS processes defining the behavior of the *clients*, *agents*, and *markets* portTypes. Due to space limitations we do not include the semi-formal behavioral representation in terms of XML code for each of the entities, but give a graphical representation of the workflows (which is directly correspondent to the XML and more readable than the XML code). In doing this we use a refinement construct which allows us to express an entire workflow as a single box in a larger workflow including it. Such a construct simply stands for replacement of the box with the content of the refinement and is used for clarity of presentation.

Figures 5, 6, and 7 contain the main diagrams describing the behaviours of the client, the agent and the market entities. Refinements boxes of the client, agent and market entities are defined in figure 8,in figure 9, and in figures 10 and 11, respectively.

5 Conclusions

An e-barter system is an e-commerce environment where agents exchange resources on behalf of their respective users, exchanges do not necessarily involve money, and agents and markets are structured in a hierarchical fashion in such a way that markets may become higher order (representative) agents. This kind of systems were formally specified in [6,7]. Unfortunately, there is a gap between the formal specification and a suitable design for the system. While the formal specification level turns out to be fundamental in defining the functional ideal behavior of the system, several practical issues were not addressed in the formal specification. In this paper we have developed a design of an e-barter system in terms of *web services*. Web services provide a suitable and well-supported model (a number of related standard and tools have been developed) for defining the behavior of e-barter systems in a distributed way. Tasks like conceptual decomposition and parallel execution of independent activities by means of different entities are implicitly performed as a result of the definition of the system as a set of web service orchestrations. In doing this, the formal specification represents the ideal behavior of the system which is to be realized by adopting adequate design (architectural) choices at the level of Web Service orchestration. By developing this case study we experienced: need of local naming technique via multiple UDDI services, introduction of new entities (such as the manager of the exchange matrix), design decisions about synchronization of re-start of cycles of exchanges in different markets,

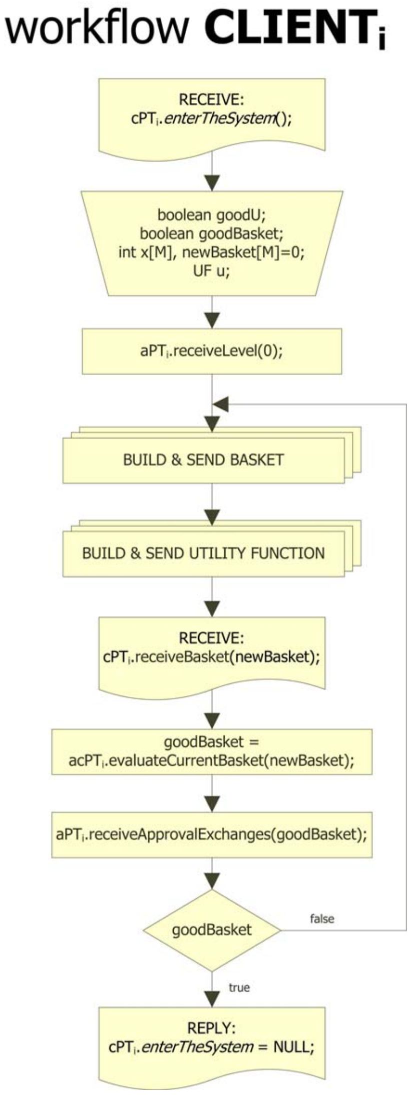

workflow CLIENTi
RECEIVE: cPTi.enterTheSystem();
boolean goodU; boolean goodBasket; int x[M], newBasket[M]=0; UF u;
aPTi.receiveLevel(0);
BUILD & SEND BASKET
BUILD & SEND UTILITY FUNCTION
RECEIVE: cPTi.receiveBasket(newBasket);
goodBasket = acPTi.evaluateCurrentBasket(newBasket);
aPTi.receiveApprovalExchanges(goodBasket);
goodBasket
false
true
REPLY: cPTi.enterTheSystem = NULL;

Fig. 5. BPEL4WS process for clients.

design decisions in structure of data exchanged and consequent tradeoff between optimality of exchanges and efficiency of the system, etc. Besides, let us note that a definition of the system in terms of web services does not only provide a suitable design, but also (partially) an implementation, since web service orchestrations defined with BPEL4WS are executable by ad-hoc interpreters.

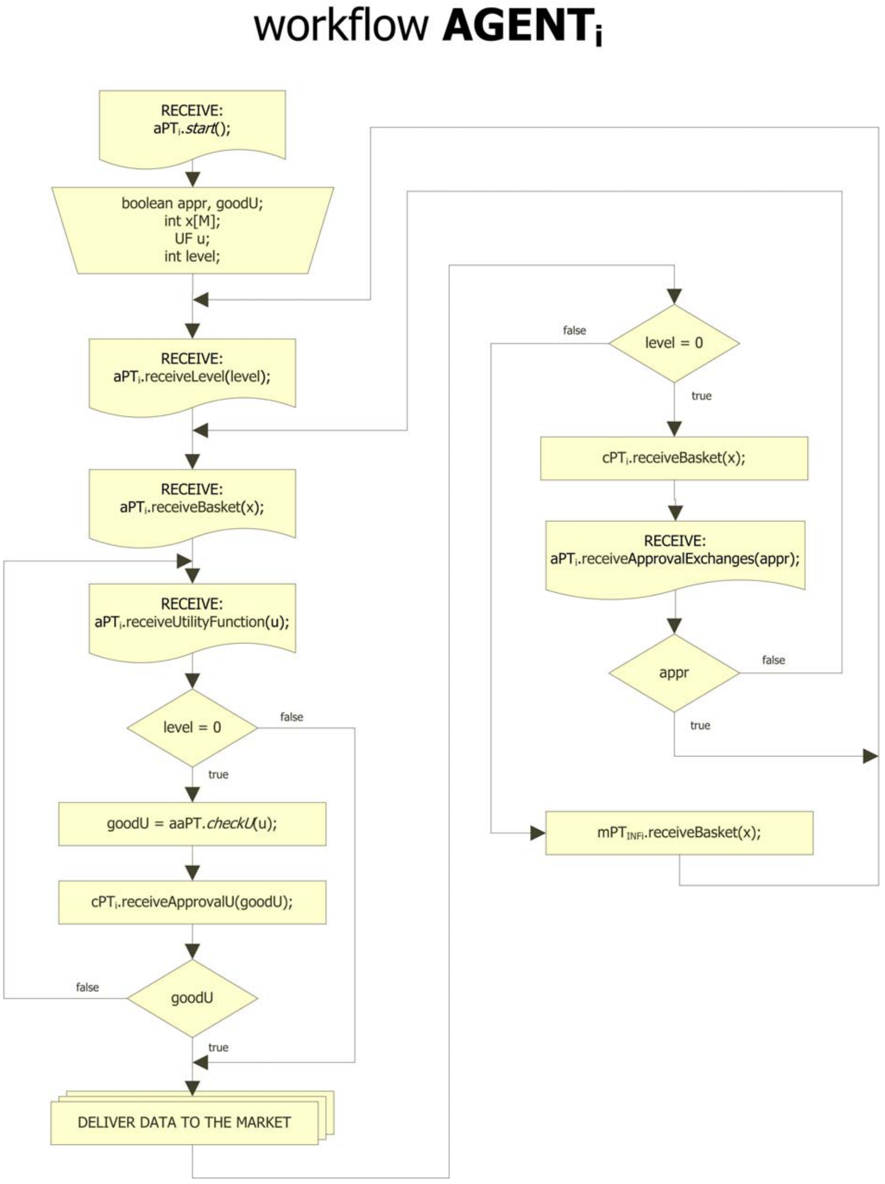

workflow AGENTi
RECEIVE: aPTi.start();
boolean appr, goodU; int x[M]; UF u; int level;
RECEIVE: aPTi.receiveLevel(level);
RECEIVE: aPTi.receiveBasket(x);
RECEIVE: aPTi.receiveUtilityFunction(u);
level = 0
false
true
goodU = aaPT.checkU(u);
cPTi.receiveApprovalU(goodU);
goodU
false
true
DELIVER DATA TO THE MARKET
level = 0
false
true
cPTi.receiveBasket(x);
RECEIVE: aPTi.receiveApprovalExchanges(appr);
appr
false
true
mPTINFi.receiveBasket(x);

Fig. 6. BPEL4WS process for agents.

References

[1] T. Andrews and F. Curbera. Web Service Business Process Execution Language, Working Draft, 2004. Version 2.0, 1.

[2] A. Cavalli and S. Maag. Automated test scenarios generation for an e-barter system. In *19th ACM Symposium on Applied Computing, SAC'04*, pages 795–799. ACM Press, 2004.

[3] E. Christenses, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL 1.1), 2001. Note 15, `http://www.w3.org/TR/wsdl`.

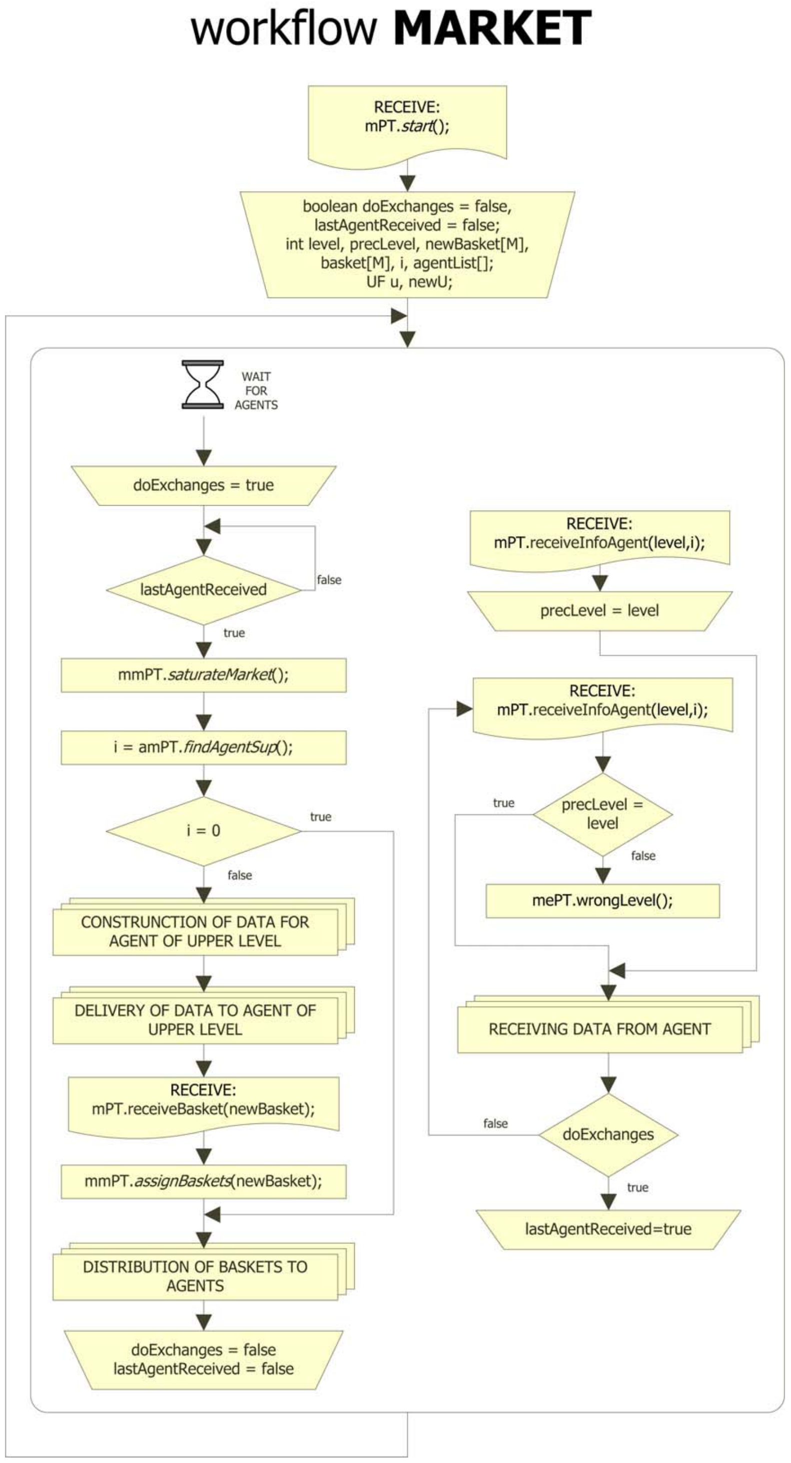

workflow MARKET
RECEIVE: mPT.start();
boolean doExchanges = false, lastAgentReceived = false; int level, precLevel, newBasket[M], basket[M], i, agentList[]; UF u, newU;
WAIT FOR AGENTS
doExchanges = true
lastAgentReceived
false
true
mmPT.saturateMarket();
i = amPT.findAgentSup();
i = 0
true
false
CONSTRUNCTION OF DATA FOR AGENT OF UPPER LEVEL
DELIVERY OF DATA TO AGENT OF UPPER LEVEL
RECEIVE: mPT.receiveBasket(newBasket);
mmPT.assignBaskets(newBasket);
DISTRIBUTION OF BASKETS TO AGENTS
doExchanges = false lastAgentReceived = false
RECEIVE: mPT.receiveInfoAgent(level,i);
precLevel = level
RECEIVE: mPT.receiveInfoAgent(level,i);
precLevel = level
true
false
mePT.wrongLevel();
RECEIVING DATA FROM AGENT
doExchanges
false
true
lastAgentReceived=true

Fig. 7. BPEL4WS process for markets.

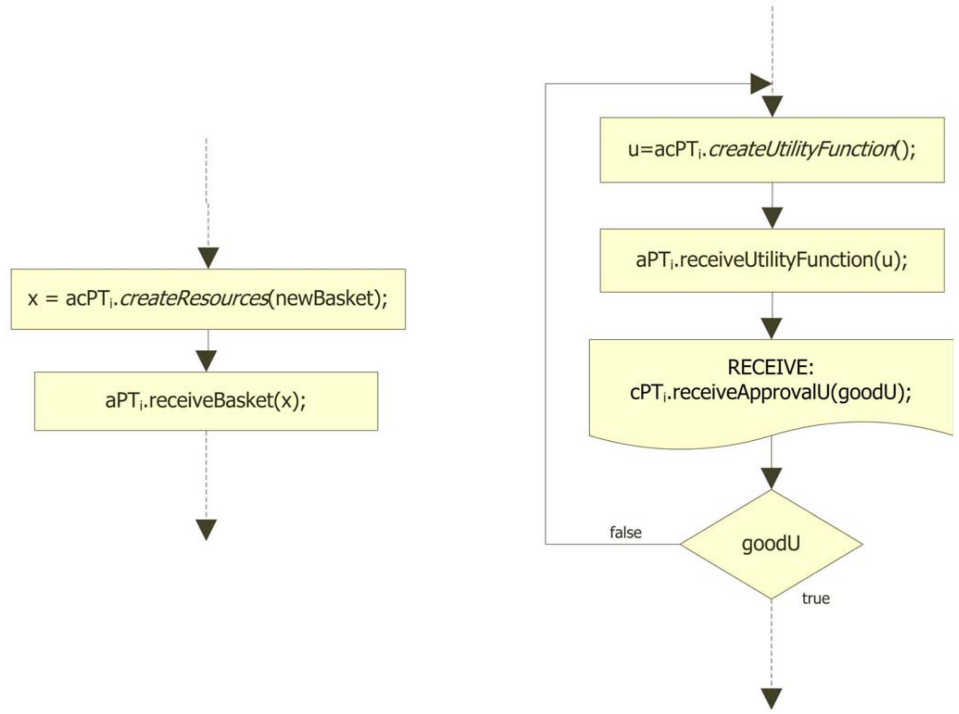

x = acPTi.createResources(newBasket);
aPTi.receiveBasket(x);
u=acPTi.createUtilityFunction();
aPTi.receiveUtilityFunction(u);
RECEIVE:
cPTi.receiveApprovalU(goodU);
goodU
false
true

Fig. 8. Refinements of the client entity: (second level) refinement of *Build & Send Basket* (left) and (second level) refinement of *Build & Send Utility Function* (right).

[4] K.V. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J.C. Meyer. Formal semantics for an abstract agent programming language. In *Intelligent Agents IV, LNAI 1365*, pages 215–229. Springer, 1998.

[5] M.N. Huhns and M.P. Singh. Service-oriented computing: Key concepts and principles. In *IEEE Internet Computing*, pages 75–81. IEEE Computer Society Press, 2005.

[6] N. López, M. Núñez, I. Rodríguez, and F. Rubio. A formal framework for e-barter based on microeconomic theory and process algebras. In *Innovative Internet Computer Systems, LNCS 2346*, pages 217–228. Springer, 2002.

[7] N. López, M. Núñez, I. Rodríguez, and F. Rubio. A multi-agent system for e-barter including transaction and shipping costs. In *18th ACM Symposium on Applied Computing, SAC'03*, pages 587–594. ACM Press, 2003.

[8] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[9] M. Núñez, I. Rodríguez, and F. Rubio. Formal specification of multi-agent e-barter systems. *Science of Computer Programming (to appear)*, 2005. in press.

[10] M. Núñez, I. Rodríguez, and F. Rubio. Specification and testing of autonomous agents in e-commerce systems. *Software Testing, Verification and Reliability*, 15(4), 2005. in press.

[11] R.L. Probert, Y. Chen, B. Ghazizadeh, P. Sims, and M. Cappa. Formal verification and validation for e-commerce: Theory and best practices. *Journal of Information and Software Technology*, 45(11):763–777, 2003.

[12] A.S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Agents Breaking Away, LNAI 1038*, pages 42–55. Springer, 1996.

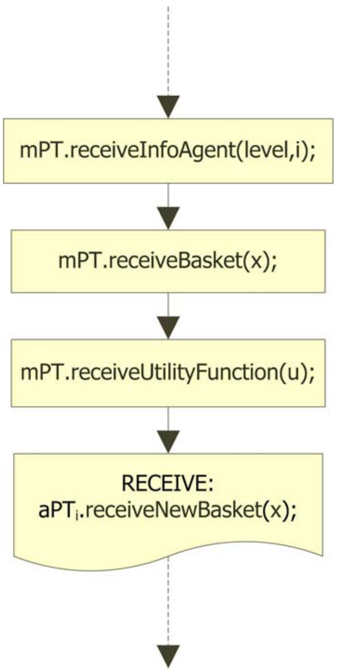

mPT.receiveInfoAgent(level,i);
mPT.receiveBasket(x);
mPT.receiveUtilityFunction(u);
RECEIVE:
aPTi.receiveNewBasket(x);

Fig. 9. Refinements of the agent entity:(Second level) refinement of *Deliver Data to the Market.*

[13] SOAP - Simple Object Access Protocol. `http://www.w3.org/TR/soap`.

[14] UDDI - Universal Description, Discovery and Integration of Web Services. `http://www.uddi.org/specification.html`.

[15] World Wide Web Consortium (W3C). `http://www.w3.org`.

6 Appendix: Operational semantics of the specification model

Next we present a brief description of the operational semantics of the specification language used to specify e-barter systems. Besides, in order to ease the presentation several details have been removed from the semantics. A full description can be found in [9], though a few details have been changed to properly represent some practical requirements that were discovered during the construction of the system design. In the next definition we present the *anchor case* of our operational semantics. In order to perform complex exchanges, agents should first indicate the barters they are willing to accept.

Definition 6.1 Let $A = (S, u, \overline{x})$ be a completed market. The *exchanges* the

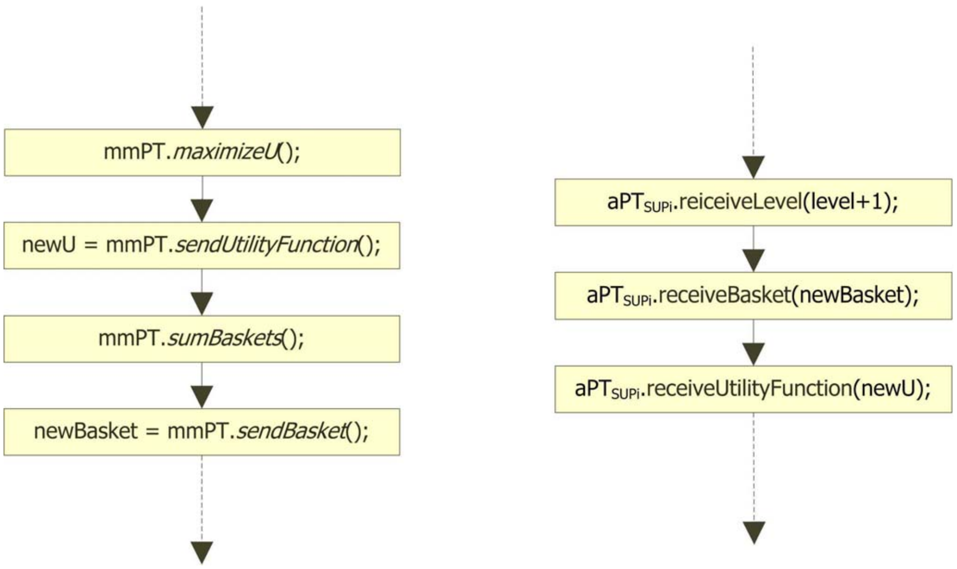

mmPT.maximizeU();
newU = mmPT.sendUtilityFunction();
mmPT.sumBaskets();
newBasket = mmPT.sendBasket();
aPT_SUPi.reiceiveLevel(level+1);
aPT_SUPi.receiveBasket(newBasket);
aPT_SUPi.receiveUtilityFunction(newU);

Fig. 10. Refinements of the market entity: (second level) refinement of *Construction of Data for Agent of Upper Level* (left) and (second level) refinement of *Delivery of Data to Agent of Upper Level* (right).

agent A would perform are given by the following operational rules:

$$\frac{u(\overline{x}+\overline{y})\geq u(\overline{x}) \wedge (\overline{x}+\overline{y})\geq\overline{0}}{(S,u,\overline{x})\xrightarrow{\overline{y}}(S,u,\overline{x}+\overline{y})}$$

$$\frac{u(\overline{x}+\overline{y})> u(\overline{x}) \wedge (\overline{x}+\overline{y})\geq\overline{0}}{(S,u,\overline{x})\xmapsto{\overline{y}}(S,u,\overline{x}+\overline{y})}$$

where $\overline{y} \in \mathbb{R}^p$, being p the number of different commodities. □

Let us remark that $\overline{y}$ may have negative components. Actually, these tuples will contain the barters offered by the agent. For example, if $\overline{y} = (1, -1, 0, -3)$ fulfils the premise then the agent would accept a barter where it is offered one unit of the first product in exchange of one unit of the second good and three units of money. Regarding the rules, the first premise simply indicates that the agent would not decrease (resp. would increase) its utility. The second premise indicates that the agent does not run into *red numbers*, that is, an agent cannot offer a quantity of an item if it does not own it. Thus, a transition as $\longrightarrow$ denotes that the agent does not worsen; a transition $\longmapsto$ denotes that the agent does improve. Next we show how offers are combined.

Definition 6.2 Let $M = \texttt{unsat}(M_1, \ldots, M_n)$ be an uncompleted market and

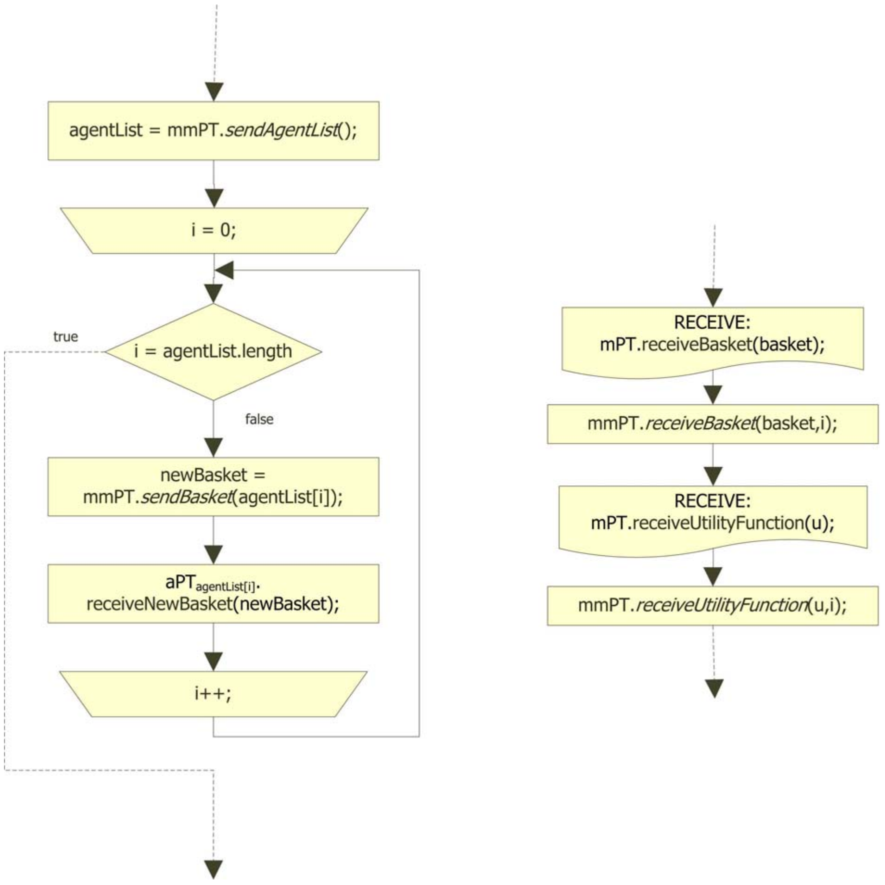

agentList = mmPT.sendAgentList();
i = 0;
true
i = agentList.length
false
newBasket = mmPT.sendBasket(agentList[i]);
aPT agentList[i].receiveNewBasket(newBasket);
i++;
RECEIVE: mPT.receiveBasket(basket);
mmPT.receiveBasket(basket,i);
RECEIVE: mPT.receiveUtilityFunction(u);
mmPT.receiveUtilityFunction(u,i);

Fig. 11. Refinements of the market process: (second level) refinement of *Distribution of Baskets to Agents* (left) and (second level) refinement of *Receiving data from Agent* (right).

$I = \{s_1, \ldots, s_r\} \subseteq \{1, \ldots, n\}$ be a set of indexes denoting the completed markets belonging to M (that is, for any $i \in I$ we have that $M_i = (S_i, u_i, \overline{x_i})$). We say that the matrix $\mathcal{E} \in (\mathbb{R}_+^p)^{n \times n}$ is a *valid exchange matrix for* M, denoted by $\texttt{valid}(M, \mathcal{E})$, if the following conditions hold:

- For any $1 \leq i \leq n$ we have $\sum_j \mathcal{E}_{ij} \leq \overline{x_i}$,
- for any $1 \leq i \leq n$ we have $\mathcal{E}_{ii} = \overline{0}$, and
- for any $1 \leq i, k \leq n$ such that $k \notin I$ we have $\mathcal{E}_{ki} = \overline{0}$ and $\mathcal{E}_{ik} = \overline{0}$.

□

First, let us note that the notion of *valid* matrix is considered only in the context of uncompleted markets: If a market is already completed then no more exchanges can be performed. Second, only completed markets belonging

$$\frac{\exists\, k \in I : M_k \xmapsto{\overline{y_k}} M'_k \;\wedge\; \forall\, i \in I, M_i \xrightarrow{\overline{y_i}} M'_i \;\wedge\; \texttt{valid}(M, \mathcal{E})}{M \overset{\mathcal{E}}{\rightsquigarrow} \texttt{unsat}(M'_1, \ldots, M'_n)}$$

where
- $M = \texttt{unsat}(M_1, \ldots, M_n)$
- $\mathcal{E} \in (\mathbb{R}^p_+)^{n \times n}$ and $I = \{s_1, \ldots, s_r\} \subseteq \{1, \ldots, n\}$
- $M'_i = \begin{cases} M_i & i \notin I \\ (S_i, u_i, \overline{x_i} + \overline{y_i}) & \text{otherwise} \end{cases}$
- $\overline{y_i} = \sum_j \mathcal{E}_{ji} - \sum_j \mathcal{E}_{ij}$, for any $i \in I$

Fig. 12. Operational rule for the exchange of resources in an uncompleted market.

to an uncompleted one may perform exchanges among them. This restriction allows to give priority to transactions performed by *closer* agents belonging to uncompleted sub-markets. Regarding the definition of *valid* matrix, the components of matrixes $\mathcal{E}$ are baskets of resources (that is, elements belonging to $\mathbb{R}^p_+$). Thus, $\mathcal{E}_{ij}$ represents the basket of resources that the market M_i would give to M_j. The condition $\sum_j \mathcal{E}_{ij} \le \overline{x_i}$ indicates that the total amount of resources given by the market M_i must be less than or equal to the basket of resources owned by that market. Finally, let us comment that an exchange does not need to include all of the completed markets. That is, if we have an exchange where only r' markets participate, then the rows and columns corresponding to the remaining $r - r'$ completed markets will be filled with $\overline{0}$. Besides, the rows and columns corresponding to the $n - r$ uncompleted markets will be also filled with $\overline{0}$.

Next we introduce the rules defining the exchange of resources. Intuitively, if we have a valid exchange matrix where at least one of the involved agents improves and no one worsens then the corresponding exchange can be performed.

Definition 6.3 Let $M = \texttt{unsat}(M_1, \ldots, M_n)$ be an uncompleted market and $I = \{s_1, \ldots, s_r\} \subseteq \{1, \ldots, n\}$ be a set of indexes denoting the completed markets belonging to M (that is, for any $i \in I$ we have that $M_i = (S_i, u_i, \overline{x_i})$). The operational transitions denoting the exchange of resources that M may perform are given by the rule shown in Figure 12. We say that M is a *local optimum*, denoted by $M \not\rightsquigarrow$, if there do not exist M' and $\mathcal{E}$ such that $M \overset{\mathcal{E}}{\rightsquigarrow} M'$. □

The operational rule presented in Figure 12 is applied under the same conditions appearing in the definition of a valid exchange matrix: It is applied to uncompleted markets and the exchange is made among a subset of the completed sub-markets. The premises indicate that at least one completed market improves after the exchange and that none deteriorates. Let us remind that, in general, a market may generate both $M_i \xrightarrow{\overline{y}} M_i'$ and $M_i \xmapsto{\overline{y}} M_i'$. So, the previous rule also considers situations where more than one market improves (we only require that at least one improves). Besides, let us remark that $M_i \xrightarrow{\overline{0}} M_i'$ always holds. So, a market not involved in the current exchange does not disallow the exchange. Regarding the conclusion, sub-markets belonging to M are modified according to the corresponding exchange matrix, while uncompleted sub-markets do not change. For completeness reasons, some other minor rules concerning the propagation of the operator $\overset{\mathcal{E}}{\rightsquigarrow}$ must be added (see [9] for further details).

If a market reaches an optimum then we need to modify the attribute of the market by replacing a term such as $\texttt{unsat}(M_1, \ldots, M_n)$ by a term such as $(S, u, \overline{x})$. Once a market is completed, resources are recursively moved from the corresponding agents to the leaves of the tree. Let us remark that a market gets completed when all of its sub-markets are completed. However, in the following we will consider it in a different way. In particular, the rule defining how a market becomes a higher order market will differ from the rule given in [9]. As we said before, in order to allow the construction of a more practical design, the behavior originally described by the formal model is slightly modified in this paper. In particular, we adapt some aspects of the formal model to the feedback provided during the development of the system design. Concretely, we allow a market to become completed even if only a subset of agents is completed. Moreover, we allow the new higher order agent to be composed from only a subset of the agents it contains. By doing so, the formal model can also represent a situation where some agents are not considered because they try to join the new higher order market after the timeout was reached. In addition, it can also represent the *alternative* approach we described before: Only those agents that do not increment their utility in the current level participate in the higher level. These situations were not be represented in the original formal model because they are motivated by some low level details that are beyond the abstraction level of the specification. However, these issues turned out to be relevant to accurately describe the overall behavior of the e-barter system in this paper. This is an example of the two-fold relation existing between the formal model and the web service design, and how both can be used to adapt each other.

Note that the modification of the formal model will consist in enabling new behaviors that were not considered in the former model. The model presented in [9] does not allow a market to become completed until all submarkets do so, and all submarkets are always included in the new agent. So, the new specification can be regarded as a *generalization* of the previous one: According to the new specification, more behaviors are possible, so the former specification is a *refinement* of the new one (which, unfortunately, diverges from our current necessities). In fact, the new model is created to enable the consistent construction of an implementation where some practical issues are dealt as we need. Note that the implementation is also a kind of (lower level) refinement. Thus, our model is generalized to allow a new (low level) refinement in a different direction.

The following rule uses two auxiliary notions: `Deliver` and `CreateUtility`. Function `Deliver` distributes a basket of resources among the original agents that are located in the leaves of the tree that is provided. On the other hand, function `CreateUtility` computes a combined utility function from the ones provided as arguments as it is described in the algorithm shown in Section 2, step 3 (see the formal definition of both functions in [9]).

Definition 6.4 Let $M = \mathtt{unsat}(M_1, \ldots, M_n)$ be an uncompleted market. Let $I = \{s_1, \ldots, s_r\} \subseteq \{1, \ldots, n\}$ where for any $i \in I$ we have $M_i = (S_i, u_i, \overline{x}_i)$. The following rule modifies the market from uncompleted to completed:

$$\frac{M \not\rightsquigarrow}{M \rightsquigarrow ([A_1, \ldots, A_r], u, \sum_{i \in I} \overline{x_i})}$$

where $u = \mathtt{CreateUtility}(u_{s_1}, \ldots, u_{s_r}, \overline{x_{s_1}}, \ldots, \overline{x_{s_n}})$ and for any $1 \leq i \leq r$ we have $A_i = (S'_i, u_{s_i}, \overline{x_{s_i}})$ with $S'_i = \mathtt{Deliver}(S_{s_i}, u_{s_i}, \overline{x_{s_i}})$. □

Let us remark that in the previous rule the transition $\rightsquigarrow$ is not labelled. These transitions play a role similar to internal transitions in classical process algebras. In order to propagate the transformations given by a $\rightsquigarrow$ transition to the context of different constructors, other minor rules have to be added. These rules can be found in [9].