Compositional Verification of a Multi-Agent System for One-to-Many Negotiation

Brazier, F.M.; Cornelissen, F.J.; Gustavsson, R.; Jonker, C.M.; Lindeberg, O.; Polak, B.; Treur, J.

published in
Applied Intelligence
2004

DOI (link to publisher)
10.1023/B:APIN.0000013334.33853.0c

document version
Publisher's PDF, also known as Version of record

Compositional Verification of a Multi-Agent System for One-to-Many Negotiation*

Frances M.T. Brazier[1], Frank Cornelissen[1], Rune Gustavsson[2], Catholijn M. Jonker[1], Olle Lindeberg[2], Bianca Polak[1], Jan Treur[1]

[1]Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl /{frances,jonker,treur} Email: {frances,jonker,treur}@cs.vu.nl

[2]University of Karlskrona/Ronneby (HK/R)
Department of Computer Science and Business Administration, Research Laboratory SIKT
URL: http:// www.sikt.hk-r.se Email: {Rune.Gustavsson, Olle.Lindeberg}@ide.hk-r.se

Abstract

Verification of multi-agent systems hardly occurs in design practice. One of the difficulties is that required properties for a multi-agent system usually refer to multi-agent behaviour which has nontrivial dynamics. To constrain these multi-agent behavioural dynamics, often a form of organisational structure is used, for example, for negotiating agents, by following strict protocols. The claim is that these negotiation protocols entail a structured process that is manageable with respect to analysis, design and execution of such a multi-agent system. In this paper this is shown by a case study: verification of a multi-agent system for one-to-many negotiation in the domain of load balancing of electricity use. A compositional verification method for multi-agent systems is applied that allows to (1) logically relate dynamic properties of the multi-agent system as a whole to dynamic properties of agents, and (2) logically relate dynamic properties of agents to properties of their subcomponents. Given that properties of these subcomponents can be verified by more standard methods, these logical relationships provide proofs of the dynamic properties of the multi-agent system as a whole.

1. Introduction

Development of agent systems in practice is often performed by a direct programming approach, providing implementation code without specification of a design at a conceptual and formal level. A first step to a principled approach to agent system development is using a structured design method (tuned to the specific area of multi-agent systems). The use of such a design method provides a conceptual design specification or design model in addition to the implementation code. The development of such design methods for agent systems is currently underway; e.g., [6], [14], [18], [26]. A design usually specifies the *structure* of an agent system. However, even if a design specification of a multi-agent system is available, it is often difficult to guarantee that this design specification actually fulfils the needs, i.e., whether it satisfies the design requirements (verification). Requirements specify the *behaviour* or *dynamics* of an agent system. Especially for critical applications, there is a need to prove that the designed system has certain behavioural or dynamic properties under certain conditions (assumptions). This means that it has to be analysed

* A shorter, preliminary version appeared in: Y. Demazeau (ed.), Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98, IEEE Computer Society Press, 1998, pp. 49-56.

how a given structure entails certain required behavioural dynamics. While developing a proof of such dynamic properties, also the assumptions that define the bounds within which the system will function properly, are generated. Research on formal verification methods specialised for agent systems is rather rare, one of the very few exceptions being [14], where verification of agent systems is addressed using a temporal belief logic.

The methodological approach followed in this paper assumes two specification languages: a language to specify *behavioural properties* or *requirements* for (systems of) agents, in addition to a language to specify *design descriptions.* Each of these languages fulfills its own purpose. A language to specify a (multi-agent) system architecture needs features different from a language to express properties of such a system. Therefore, in principle the two languages are different. The distinction between these specification languages follows the distinction made in the AI and Design community (cf. [15]) between the *structure* of a design object on the one hand, and *function* or *behaviour* on the other hand. Formal models specified in the two languages can be related in a formal manner: it is formally defined when a design description satisfies a behavioural property specification, and this formal relation is used to verify that the design description fulfills the requirements.

Therefore, in addition to a multi-agent system design method based on a design specification language, a method and language for formal analysis, i.e., for requirements engineering and verification have to be developed. In contrast to previous papers addressing design specification, such as [6], (for a case study in design specification, see [6]), this paper addresses, for a real-world case, specification of behavioural properties, identification of logical relations between these properties, and verification of these properties for a design specification.

For nontrivial examples, verification can be a very complex process, both in the conceptual and computational sense. For these reasons, a recent trend in the literature on verification in general is to exploit compositionality and abstraction to structure the process of verification; cf. [1], [17], [20]. The notion of compositional verification by itself is certainly not novel, but the specialisation of this notion to multi-agent systems is. In the approach presented below, the use of a compositional verification method for multi-agent systems (cf. [20]) is explored for the formal analysis of a multi-agent system for one-to-many negotiation, in particular for load balancing of electricity use; see [5]. In short, the behavioural properties of the whole system are established and logically related to assumptions that themselves are behavioural properties of agents, which in turn are related to assumptions on sub-components of agents, and so on. The behavioural properties are formalised in terms of temporal semantics.

The multi-agent system design analysed in this paper has been specified using the component-based design method for multi-agent systems DESIRE; cf. [6]. Using the DESIRE software environment, from this design specification in an automated manner an executable prototype implementation can be (and actually has been) generated by a standard implementation generator available within the software environment. Note that the formal analysis addressed in this paper establishes relationships between different formally specified behavioural properties and between formally specified behavioural properties and the design specification, and no direct relationships with the implementation code (which would be much less transparent). However, the relationships between design specification and prototype implementation code by means of the standard implementation generator at least defines an indirect relationship. The advantage of this two-step indirectness is that both design structures and behavioural properties can be specified (and related) at a conceptual level instead of an implementation level, which makes the verification process more

transparent compared to the case that the implementation code is directly addressed in a formal analysis.

The paper is structured as follows. In Section 2 the component-based design method for multi-agent systems DESIRE is briefly described, and in Section 3 the compositional verification method is introduced. Section 4 discusses the approach to one-to-many negotiation processes used, and in Section 5 the design model of the multi-agent system is briefly summarized. Section 6 addresses verification at the top level of the system, and Section 7 verification at the highest process abstraction level within one of the agents: the Utility Agent. In Section 8 the results are discussed.

2. Component-Based Design of Multi-Agent Systems

The example multi-agent system described in this paper has been developed using the component-based design method DESIRE for multi-agent systems (DEsign and Specification of Interacting REasoning components); cf. [5a]. The design at a conceptual level of a multi-agent system is supported by graphical design tools within the DESIRE software environment. Translation to an operational system is straightforward; the software environment includes implementation generators with which formal specifications can be translated into executable code of a prototype system. In DESIRE, a design model or design *structure* consists of specifications of the following three types: process composition, knowledge composition, the relation between process composition and knowledge composition. Specification in DESIRE focusses on the design description of a system; specification of behavioural properties as needed for Requirements Engineering and verification is not particularly supported within DESIRE. The three types of specification in a design description are discussed in more detail below. Notice that, since this paper addresses formal analysis of multi-agent systems, and not design specification, the necessary notions on design are only briefly sketched. For more details about design specification, see the references [6].

2.1. Process Composition

Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of (is composed of) lower level processes.

2.1.1. Identification of Processes at Different Levels of Abstraction

Processes can be described at different levels of abstraction; for example, the process of the multi-agent system as a whole, processes defined by individual agents and the external world, and processes defined by task-related components of individual agents. The identified processes are modelled as *components*. For each process the *input and output information types* are modelled. The identified levels of process abstraction are modelled as *abstraction/specialisation relations* between components: components may be *composed* of other components or they may be *primitive*. Primitive components may be either reasoning components (i.e., based on a knowledge base), or, components capable of performing tasks such as calculation, information retrieval, optimisation. These levels of process abstraction provide process hiding at each level.

2.1.2. Composition of Processes

The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *composition*. This composition of processes is described by a specification of the possibilities for *information exchange* between processes (*static view* on the

composition), and a specification of *task control knowledge* used to control processes and information exchange (*dynamic view* on the composition).

2.2. Knowledge Composition

Knowledge composition identifies the knowledge structures at different levels of (knowledge) abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction levels may correspond to the process abstraction levels, but this is often not the case.

2.2.1. Identification of Knowledge Structures at Different Abstraction Levels

The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. Knowledge structures can be identified and described at different levels of abstraction. At higher levels details can be hidden. An *information type* defines an ontology (lexicon, vocabulary) to describe objects or terms, their sorts, and the relations or functions that can be defined on these objects. Information types can logically be represented in order-sorted predicate logic. To illustrate these notions a simple example is used. In this example agent A (with role service request generation) communicate a (service) request to agent B (with role service provision). Agent B answers with a service proposal, which is accepted by agent A. In this example the information type for the *input* of agent A is:

sorts

AGENT, REQUEST, PROPOSAL

objects

r1: REQUEST

p1: PROPOSAL

relations

proposal_for_from: PROPOSAL x REQUEST x AGENT x AGENT

An example of an information type is, for the *output* of an agent A with role service request generation (notice that more objects in the sorts can be defined):

sorts

AGENT, REQUEST, PROPOSAL

objects

r1: REQUEST

p1: PROPOSAL

relations

request_for_from: REQUEST x AGENT x AGENT

accepted_proposal_for_from: PROPOSAL x REQUEST x AGENT x AGENT

An example information type for the *input* of an agent B is:

sorts

AGENT, REQUEST, PROPOSAL

objects

r1: REQUEST

p1: PROPOSAL

relations

request_for_from: REQUEST x AGENT x AGENT

accepted_proposal_for_from: PROPOSAL x REQUEST x AGENT x AGENT

An example information type for the *output* of agent B (with role service provision) is:

sorts

AGENT, REQUEST, PROPOSAL

objects

r1: REQUEST

p1: PROPOSAL

relations

proposal_for_from: PROPOSAL x REQUEST x AGENT x AGENT

An example *internal* information type for agent B (with role service provision) is:

sorts

REQUEST, PROPOSAL

objects

r1: REQUEST

p1: PROPOSAL

relations

qualified_proposal_for: PROPOSAL x REQUEST

These information types specify an ontology, for example, for a request R of agent A to agent B (request_for_from(R, A, B)), a proposal P of B to A to satisfy a request R (proposal_for_from(P, R, B, A), and a citerion to indicate whether a proposal qualifies to fulfill a request (qualified_proposal_for(P, R)).

A *knowledge base* defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented by formulae in order-sorted predicate logic, which can be normalised by a standard transformation into rules. An example of a part of a knowledge base (of agent B) is:

qualified_proposal_for(p1, r1).

if request_for_from(R:REQUEST, B:AGENT, A:AGENT)
and qualified_proposal_for(P:PROPOSAL, R:REQUEST)
then proposal_for_from(P:PROPOSAL, R:REQUEST, A:AGENT, B:AGENT).

This expresses that p1 is a qualified proposal for r1, and if any request R is done by agent B to agent A, for which P is a qualified proposal, then this P is a proposal from agent A for request R to agent B.

2.2.2. Composition of Knowledge Structures

Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of more specific

knowledge bases. The compositional structure is based on the different levels of knowledge abstraction distinguished, and results in information and knowledge hiding.

2.3. Relation between Process and Knowledge Composition

Each process in a process composition uses knowledge structures. Which knowledge structures are used for which processes is defined by the relation between process composition and knowledge composition.

2.4 Trace Semantics of a Design

Semantics of a component-based design is based on a set of traces of three-valued information states. An *information state* I of a system or system component D (e.g., the overall system, or an input or output interface of an agent) is an assignment of truth values {true, false, unknown} to the set of ground atoms describing the information within D (i.e., based on the interface information types). The set of all possible information states of D is denoted by IS(D). A *trace* M of D is a sequence (over the natural numbers) of information states $(I_t)_{t \in \mathbf{N}}$ in IS(D). Each design defines such a set of traces. For more details, see [7].

3. Compositional Verification

The purpose of verification is to formulate relevant functional or behavioural properties of a system (and its components), and to prove that, under a certain set of conditions (assumed properties), a system will adhere to a certain set of desired properties (for example the behavioural requirements for the design). Compositional verification, in particular, is a well-known method to verify systems based on concurrently processing components; cf [23]. In the compositional verification approach for multi-agent systems used in this paper (and adopted from [20]), this is done by a mathematical proof (i.e., a proof in the form mathematicians are accustomed to do) that the specification of the system together with the assumed properties implies the behavioural properties that it needs to fulfil.

3.1. The Compositional Verification Method

A component-based multi-agent system can be viewed at different levels of process abstraction. Viewed from the top level, denoted by L_0, the complete system is one component S; internal information and processes are hidden. At the next, lower level of abstraction, the system component S can be viewed as a composition of agents and the world. Each agent is composed of its sub-components, and so on. The compositional verification method takes this compositional structure into account. Verification of a composed component is done using:

- properties of the *sub-components* it embeds,
- the way in which the component is composed of its sub-components (the *composition relation*),
- *environmental properties* of the component (depending on the rest of the system, including the world)

Given the specification of the composition relation, the assumptions under which the component functions properly are the environmental properties and the properties to be proven for its sub-components. This implies that properties at different levels of process abstraction are involved in

the verification process. The primitive components (those that are not composed of other components) can be verified using more traditional verification methods; for an overview, see, e.g., [21]. Often the properties involved are not given at the start: to find them is one of the aims of the verification process.

The verification proofs that connect properties of one process abstraction level with properties of the other level are compositional in the following manner: any proof relating level *i* to level *i+1* can be combined with any proof relating level *i-1* to level *i*, as long as the same properties at level *i* are involved. This means, for example, that the whole compositional structure beneath level *i* can be replaced by a completely different design as long as the same properties at level *i* are achieved. After such a modification only the proof for the new component has to be provided. In this sense the verification method supports reuse of verification proofs. The compositional verification method can be formulated as follows:

A. Verifying one Level Against the Other

For each abstraction level the following procedure for verification is followed:

1. Determine which properties are of interest (for the higher level).
2. Determine which assumed properties (at the lower level) are needed to guarantee the properties of the higher level, and which environment properties.
3. Prove the properties of the higher level on the basis of these assumed properties, and the environment properties.

B. Verifying a Primitive Component

For primitive knowledge-based components a number of techniques exist in literature, see for example [21]. For primitive non-knowledge-based components, such as databases, or neural networks, or optimisation algorithms, verification techniques can be used that are especially tuned for that type of component.

C. The Overall Verification Process

To verify the entire system

1. Determine the properties that are desired for the whole system.
2. Apply **A** iteratively. In the iteration the desired properties of each abstraction level L_i are the assumed properties for the higher level.
3. Verify the primitive components according to **B**.

Notes:

- The results of verification are two-fold:
  (1) Properties at the different abstraction levels.
  (2) The logical relations between the properties of adjacent abstraction levels.
- process and information hiding limits the complexity of the verification per abstraction level.
- a requirement to apply the compositional verification method described above is the availability of an explicit specification of how the system description at an abstraction level L_i is composed from the descriptions at the lower abstraction level L_{i+1}; the component-based design method for multi-agent systems DESIRE fulfils this requirement.
- in principle different procedures can be followed (e.g., top-down, bottom-up or mixed).

3.2. Language and Semantics used

In contrast to a design specification language as in DESIRE, which specifies design structure, a language is needed to specify dynamic properties of an agent system's behaviour. To obtain a formalisation of behavioural properties different variants of temporal logic can be used, depending on the type of properties to be expressed. For example, linear or branching time temporal logic are appropriate to specify various agent (system) behavioural properties. Examples of the use of specification languages based on such variants of temporal logic are described, for example in [11], [14], [22]. However, to specify adaptive properties such as 'exercise improves skill' as well, a comparison between different histories has to be explicitly expressed. This requires a form of temporal logic language which is more expressive than those allowing to refer at each time point only to one history. An example of such a more expressive formal language in which different histories can be compared is the Temporal Trace Language TTL introduced in [20]; this language is defined as follows.

A structure consisting of a number of component names, a sub-component relation, and interface information types for each component is assumed given. As in Section 2.4 an *information state* I of a component D (e.g., the overall system, or an input or output interface of an agent) is an assignment of truth values {true, false, unknown} to the set of ground atoms describing the information state within D. The set of all possible information states of D is denoted by IS(D). A *trace* $\mathbf{M}$ of D is a sequence (over the natural numbers) of information states $(I_t)_{t \in \mathbf{N}}$ in IS(D). Given a trace $\mathbf{M}$ of D, the information state of the input interface of an agent A at time point t is denoted by

$$state_D(\mathbf{M}, t, input(A)),$$

where $state_D$ and input are function symbols. Analogously,

$$state_D(\mathbf{M}, t, output(A))$$

denotes the information state of the output interface of agent A at time point t within system (component) D. The information states can be related to statements via the formally defined satisfaction relation denoted by the symbol |=, which has some similarity to the Holds-predicate in the Situation Calculus. Differences from the Situation Calculus approach are, however, that we

(1) use an infix notation for the |= predicate instead of a prefix notation,
(2) refer to a trace and time point instead of a single state, and
(3) can focus on part of the system.

Based on these statements, behavioural properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts **T** for time points, **Traces** for traces and **F** for state formulae, using quantifiers over time and the usual first order logical connectives such as ¬ (not), ∧ (and), ∨ (or), ⇒ (implies) , ∀ (for all), ∃ (there exists). As the language is a first-order predicate logic language, standard semantics and proof calculus can be adopted.

An example of such a statement is the following (other examples can be found in Sections 6 and 7 below). Consider the following informally expressed property for the dynamics of a multi-agent system as a whole:

> *Each service request of agent A to agent B must be followed by a service proposal of agent B after a certain time which is acknowledged as satisfactory by agent A..*

In a structured, semiformal manner, this property can be reformulated (and detailed) as follows:

> *if at some point in time*
> *agent A outputs: a service request for B,*
> *then at a later point in time*

agent B outputs: *a service proposal for the request for A*
and at a still later point in time
agent A outputs: *the proposal is accepted to B*

Using the formal language introduced above the following temporal formalisation is made of this example property:

$$\forall M, t, r \; [\; state(M, t, output(A)) \models request\_for\_from(r, B, A)$$
$$\Rightarrow [\; \exists p, t1 \geq t \;\; state(M, t1, output(B)) \models proposal\_for\_from(p, r, A, B)$$
$$\wedge \;\; \exists t2 \geq t1 \;\; state(M, t2, output(A)) \models accepted\_proposal\_for\_from(p, r, A, B)\;]\;]$$

Here the statement $state(M, t, output(A)) \infty request\_for\_from(r, B, A)$ means that within trace M at time point t a statement $request\_for\_from(r, B, A)$ occurs in the output interface of agent A, i.e. has truth value true in the output state of A.

4. One-to-many Negotiation Processes

In this section the application domain is briefly sketched, and the one-to-many negotiation process devised within this domain is presented.

4.1. Load Balancing of Electricity Use

The purpose of load management of electricity use is to smoothen peak load by managing a more appropriate distribution of the electricity use among consumers. Flexible pricing schemes can be an effective means to influence consumer behaviour; cf. [16]. The assumption behind the model presented in this paper is that, to acquire a more even distribution of electricity usage in time, consumer behaviour can be influenced by financial gain. Consumers are autonomous in the process of negotiation: each individual consumer determines which price/risk he/she is willing to take and when. As consumers are all individuals with their own characteristics and needs (partially defined by the type of equipment they use within their homes), that vary over time, models of consumers used to design systems to support the consumer, need to be adaptive and flexible (cf. [2]). Utility companies negotiate price in a one-to-many negotiation process with each and every individual separately, unaware of the specific models behind such systems for individuals. In the model discussed in this paper the negotiation process is modelled for one utility company and a number of consumers, each with their own respective agent to support them in the negotiation process: one Utility Agent and a number of Customer Agents.

4.2. Modelling the Negotiation Process

In [24] a number of mechanisms for negotiation are described. A protocol with well-defined properties, called the *monotonic concession protocol*, is described: during a negotiation process all proposed deals must be equally or more acceptable to the counter party than all previous deals proposed. The strength of this protocol is that the negotiation process always converges. The monotonic concession protocol has been applied to the load management problem, to obtain a model for the one-to-many negotiation process between one Utility Agent and a (in principle large) number of Customer Agents.

In this model, the Utility Agent always initiates the negotiation process, as soon as a coming peak in the electricity consumption is predicted. In the method used the Utility Agent constructs a

so-called *reward table* and communicates this table to all Customer Agents (*announcement*). A reward table (for a given time interval) consists of a list of possible cut-down values, and a reward value assigned to each cut-down value. The cut-down value specifies an amount of electricity that can be saved (expressed in percentages) and the reward value specifies the amount of reward the Customer Agent will receive from the Utility Agent if it lowers its electricity consumption by the cut-down value. A Customer Agent examines and evaluates the rewards for the different cut-down values in the reward tables. If the reward value offered for the specific cut-down is acceptable to the Customer Agent, it informs the Utility Agent (*bid*) that it is prepared to make a cut-down x, which may be zero to express that no cut-down is offered.

As soon as the Customer Agents have responded to the announcement of a reward table, the Utility Agent predicts the new balance between consumption and production of electricity for the stated time interval. The Utility Agent is satisfied by the responses if a peak can be avoided if all Customer Agents implement their bids. If the Utility Agent is not satisfied by the responses communicated by the Customer Agents, it announces a new reward table (according to the monotonic concession protocol mentioned above) to the Customer Agents in which the reward values are at least as high, and for some cut-down values higher than in the former reward table (determined on the basis of, for example, the formulae described in Section 5 below). The Customer Agents react to this new announcement by responding with a new bid or the same bid again (in line with the rules of the monotonic concession protocol). This process continues until (1) the peak is satisfactorily low for the Utility Agent (at most the capacity of the utility company), or (2) the reward values in the new reward table have (almost) reached the maximum value the Utility Agent can offer. This value has been determined in advance. For more details on this negotiation method, see [5].

5. Component-based Design of the Prototype System

The prototype Multi-Agent System has been fully specified and (automatically) implemented in the DESIRE software environment. The top level composition of the system consists of a Utility Agent, two Customer Agents, and an External World. The top level process composition of the system is shown in Figure 1 (picture taken from the graphical design tool within the DESIRE software environment).

5.1. Top Level Composition of the Utility Agent

The first level composition within the Utility Agent is depicted in Figure 2 (taken from the graphical design tool within the DESIRE software environment). This picture shows part of the graphical interface of the DESIRE software environment; in addition, interfaces to the agents have been implemented which are specific for this prototype (see [5]).

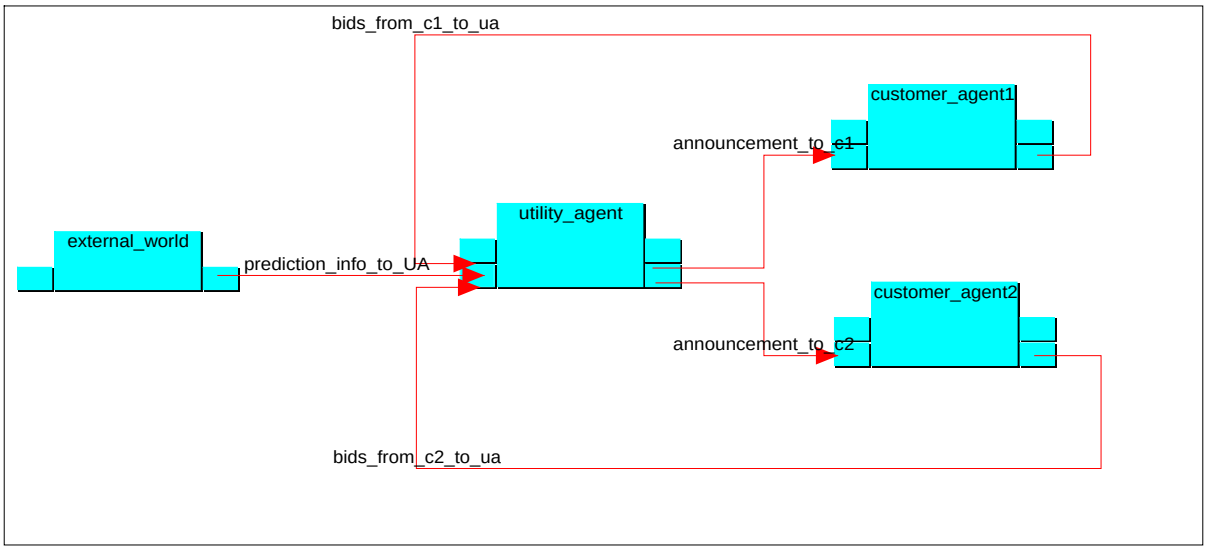

Figure 1. Process composition at the top level of the system

5.2. Knowledge used within the Utility Agent

In this prototype system the Utility Agent communicates the same announcements to all Customer Agents, in compliance with Swedish law. The predicted balance between the consumption and the production of electricity, is determined by the following formulae (here CA is a variable ranging over the set of Customer Agents):

The first of the formulae determines the prediction of a consumer CA's electricity use, after this consumer has committed to a reduction by cutdown(CA). Here predicted_use(CA) is the electricity use of customer CA during the considered period if no reduction (cutdown) is decided by CA. Moreover, cutdown(CA) denotes the reduction fraction of CA, and allowed_use(CA) is the maximal allowed use as agreed in the general contract with the customer.

predicted_use_with_cutdown(CA) = min { predicted_use(CA), (1 - cutdown(CA))* allowed_use(CA) }

The second formula takes the sum over all consumers of the difference between predicted use (assuming the reduction to which they committed) and normal use (the overall use that is considered to be optimal by the Utility Agent) to determine the predicted_overuse. This predicted_overuse is the number that needs to be reduced to zero by the negotiation process. The last formula normalises this overuse by normalising it with respect to normal use.

$$\text{predicted\_overuse} = \Sigma_{CA}\ \text{predicted\_use\_with\_cutdown(CA)} - \text{normal\_use}$$

$$\text{overuse} = \text{predicted\_overuse/normal\_use}$$

In the prototype system the increase of rewards in announcements during the negotiation process is based on the following formula

new_reward = reward + beta * overuse * (1 - reward/max_reward)* reward

Here max_reward is the maximal reward the utility company is willing to pay, and beta is a factor by which the negotiation speed can be fine-tuned. Note that the increase of rewards is proportional to the relative overuse. Therefore, if the overuse decreases, also the increases in rewards decrease during the negotiation process. The factor beta determines how steeply the reward values increase; in the current system it has a constant value. As said, the reward value increases more when the predicted overuse is higher (in the beginning of the negotiation process) and less if the predicted overuse is lower. However, the rewards never exceed the maximal reward, due to the logistic factor

(1 - reward/max_reward)

The negotiation process ends when the difference between the new reward values and the (old) reward values is less than or equal to 1. Note that for the predicted use of a customer there is no need to use an individual value: an average value based on available customer statistics is sufficient, since in the formula for predicted over-use the sum is taken over all customers. Furthermore, the predictions assume that a customer commits to the reduction as promised. To assure that customers indeed live up to these commitments, for example, high financial penalties can be used if commitments are violated.

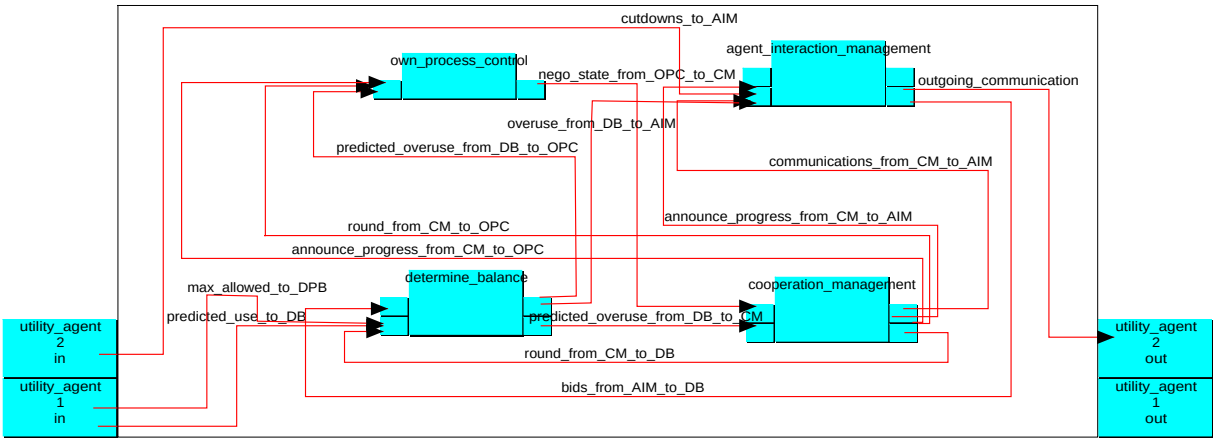

Figure 2 Process composition at the first level within the Utility Agent

6. Verification Starting at the Top Level

Two important assumptions behind the system are: energy use is (statistically) predictable at a global level, and consumer behaviour can be influenced by financial gain. These assumptions imply that if the financial rewards (calculated on the basis of statistical information) offered by a Utility Agent are well chosen, Customer Agents will respond to such offers and decrease their use.

To verify the system a top-down compositional verification (see Section 3) approach is followed. First, in Section 6.1 requirements for the system S as a whole are identified, in the form of dynamic properties. Next, in Section 6.2 requirements for the different agents (Utitility Agent (UA) and Customer Agents (CA)) and the External World component within the system are identified that

together entail the requirements of S as a whole. In Section 6.3 logical relationships are identified between these dynamic properties at different aggregation levels within the system.

The most important properties as requirements for the load balancing system S as a whole are that

(1) the negotiation process satisfies the monotonic concession protocol,
(2) at some point in time the negotiation process will terminate, and
(3) the agents make rational decisions during the negotiation process.

These properties are formally defined in Section 6.1.

To prove properties such as these, several other properties of the participating agents (and the external world) are assumed. These properties of agents and the external world are defined in Section 6.2. An important property for the Utility Agent, for example, is that after the negotiation process the predicted overuse has decreased to such an extent that is at most the maximal overuse the utility company considers acceptable. Some of the logical relationships (and proofs thereof) between properties are briefly presented in Section 6.3. Next, Section 7 shows how these assumed properties of the agents in the system can be logically related to (and proven from) properties assumed for the subcomponents of the agents, and finally in Section 8 it is shown how basic properties of a (primitive) subcomponent can be related to the subcomponent's input-output relation or knowledge base.

The properties defined at the level of the entire system are based on combinations of properties of the agents. The formalisation of the properties is done in the Temporal Trace Language TTL briefly described in Section 3.2; see also [20]. In these properties, and the properties in subsequent sections the following language elements are used

Table 1 Overview of variables and predicates from the state ontologies used in Section 6

variables	*meaning*
t, t'	Range over the values within the time frame
CD, CD'	Range over cutdown fraction values (0.1, 0.2, 0.3, …, 0.9, 1.0)
N, N'	Range over negotiation round values (natural numbers)
R, R'	Range over reward values (taken as natural numbers)
U, U'	Range over use values (real numbers)
CA	Ranges over Customer Agent names
predicates	*meaning*
round(N)	Denotes that negotiation round N occurs (a counter to indicate which negotiation round is in effect).
announcement(CD, R, N)	Denotes that in round N the utility agent offers for cutdown fraction CD a reward R.
cutdown(CD, N).	denotes that the customer agent (CA) offers a cutdown of CD in round N.
cutdown_from(CD, CA, N)	Within the Utility Agent an additional argument is added to the predicate cutdown: the ternary predicate cutdown_from to indicate the Customer Agent CA from which the cutdown originates.
predicted_overuse(U, N)	Denotes that for round N the predicted overuse is U

6.1 Properties of the System as a Whole

S1. Monotonicity of negotiation
The system S satisfies *monotonicity of negotiation* if the Utility Agent satisfies montonicity of announcements and all Customer Agents satisfy monotonicity of bids. This is formally defined as the conjunction of the Utility Agent announce monotonicity property U7 and for each Customer Agent the bid monotonicity property C5 (see below).

S2. Termination of negotiation
The system S satisfies *termination of negotiation* (on a given time interval) if a time point exists after which no announcements or bids (referring to the given time interval) are generated by the agents. This is formally defined as the conjunction of the Utility Agent termination property U0 and for each Customer Agent the termination property C0 (see below).

S3. Rationality of negotiation
The system S satisfies *rationality of negotiation* if the Utility Agent satisfies announcement rationality and each Customer Agent satisfies bid rationality. This is formally defined as the conjunction of the Utility Agent *announcement rationality* property U9 and for each Customer Agent the *bid rationality* property C4 (see below).

To be able to prove successfulness of the negotiation certain assumptions have to be made on how the 'willingness' of the Customer Agents compares to the 'willingness' of the Utility Agent. The following system property expresses that the Utility Agent is willing to offer at least or more than the required reward by the Customer Agents.

S4. Required reward limitation

The system S satisfies *required reward limitation* if for each Customer Agent and each cut-down fraction CD, the required reward of the Customer Agent $rr_{CA}(CD)$ is at most the maximal reward $mr_{UA}(CD)$ that can be offered by the Utility Agent:

$\forall CA\ \forall CD \quad rr_{CA}(CD) \leq mr_{UA}(CD)$

S5. Communication Successfulness
A system property which often is used is communication successfulness: if an agent A talks to another agent B, then agent B will hear what is said. In particular this property is needed for communication on the negotiation round, on announcements and on cut-downs.

Communication successfulness from UA to CA:

$\forall M \in Traces(S)\ \forall CA, t, X$

$state_{UA}(M, t, output(UA)) \models X \quad \Rightarrow \exists t' \geq t \quad state_{CA}(M, t', input(CA)) \models X$

This property expresses that if X occurs at the output of the Utility Agent UA, then at a later point in time it will occur at the input of the Customer Agent CA; here X can be one of round(N), announcement(CD, R, N).

Communication successfulness from CA to UA:

$\forall M \in$ Traces(S) $\forall$ CA, t, CD, N

$\text{state}_{CA}(M, t, \text{output(CA)}) \models \text{cutdown(CD, N)} \Rightarrow \exists t' \geq t \;\; \text{state}_{UA}(M, t', \text{input(UA)}) \models \text{cutdown\_from(CD, CA, N)}$

Note that within the Utility Agent an additional argument is added to the predicate cutdown: it is mapped onto the ternary predicate cutdown_from to indicate the Customer Agent CA from which the cutdown originates.

S6. Communication Groundedness

Communication successfulness only guarentees that an agent hears what is said. Sometimes also the reverse needs to be guaranteed: that an agent does not hear things that were not said. This property is formulated for communication on the negotiation round, on announcements and on cut-downs.

Communication groundedness from UA to CA:

$\forall M \in$ Traces(S) $\forall$ CA, t, X

$\text{state}_{CA}(M, t, \text{input(CA)}) \models X \Rightarrow \exists t' \leq t \;\; \text{state}_{UA}(M, t', \text{output(UA)}) \models X$

This property expresses that if X occurs at the input of the Customer Agent CA, then at an earlier point in time it has occured at the output of the Utility Agent UA; here X can be one of round(N), announcement(CD, R, N).

Communication groundedness from CA to UA:

$\forall M \in$ Traces(S) $\forall$ CA, t, CD, N

$\text{state}_{UA}(M, t, \text{input(UA)}) \models \text{cutdown\_from(CD, CA, N)} \Rightarrow \exists t' \leq t \;\; \text{state}_{CA}(M, t', \text{output(CA)}) \models \text{cutdown(CD, N)}$

The above three properties S4, S5 and S6 are assumptions for the whole system; they are assumed and used in the proofs of a number of properties.

In addition to these properties a global *successfulness property* for the whole negotiation process could be defined. However, as successfulness depends on the perspective of a specific agent, the choice has been made to define succesfulness as a property of an agent (e.g., property U1 below).

6.2 Properties of the Agents and the World

The properties of the Utility Agent, the Customer Agents, and the External World are defined in this section. Note that each of the properties is presented as a temporal statement either about all traces of the system S (i.e., U0, U1, U11, C0 below) or about all traces of the agent itself. In the latter case the truth of the property does not depend on the environment of the agent, but only on the agent's internal processes. Section 6.3 discusses how the various properties are logically related.

6.2.1 Properties of the Utility Agent

First, two properties U0 and U1 of the agent UA are defined that depend both on the agent's internal functioning and on the environment of the agent within the system as a whole. These

properties are expressed for traces $M \in$ Traces(S) of the system as a whole. Next the properties U2, U3, U4, U5, U6, U7, U8, U9, U10, and U11 are defined that only depend on the agent's internal functioning, and not on the agent's environment within the system. These properties are expressed for traces $M \in$ Traces(UA) of the agent UA independent of the rest of the system.

U0. Finite termination of negotiation by UA

The Utility Agent satisfies *finite termination of negotiation* if a round number N exists such that UA does not negotiate for any round N' after this round number, i.e., no announcement or round information are communicated for such N':

$$\forall M \in \text{Traces(S)}\ \exists N\ \forall t, CD, R, N'>N\ \ \text{state}_S(M, t, \text{output(UA)}) \not\models \text{announcement(CD, R, N')}\ \&\ \text{state}_S(M, t, \text{output(UA)}) \not\models \text{round(N')}$$

U1. Successfulness of negotiation

The Utility Agent satisfies *successfulness of negotiation* if:

at some t and for some negotiation round N the predicted overuse is less than or equal to 0

Formally:

$$\forall M \in \text{Traces(S)}\ \exists t, N\ \exists U \leq 0\quad \text{state}_S(M, t, \text{output(UA)}) \models \text{predicted\_overuse(U, N)}$$

U2. Negotiation round generation effectiveness

The Utility Agent satisfies *negotiation round generation effectiveness* if the following holds:

if and when within round N predicted overuse is positive, a next negotiation round N+1 is initiated

Formally:

$$\forall M \in \text{Traces(UA)}\ \forall t, N, U, CD, R$$
$$[\ \text{state}_{UA}(M, t, \text{output(UA)}) \models \text{round(N)}$$
$$\&\ \text{state}_{UA}(M, t, \text{output(UA)}) \models \text{predicted\_overuse(U, N)}$$
$$\&\ U > 0$$
$$\&\ \text{state}_{UA}(M, t, \text{output(UA)}) \models \text{announcement(CD, R, N)}$$
$$\&\ R < mr_{UA}(CD)\ ]$$
$$\Rightarrow\ \exists t' > t\ \text{state}_{UA}(M, t', \text{output(UA)}) \models \text{round(N+1)}$$

Here round(N+1) denotes that the Utility Agent has declared round N+1 active.

U3. Negotiation round generation groundedness

The Utility Agent satisfies *negotiation round generation groundedness* if the following holds:

if the predicted overuse is not positive, then no new negotiation round is initiated

Formally:

$$\forall M \in \text{Traces(UA)}\ \forall t, N, U$$

$state_{UA}(M, t, output(UA)) \models predicted\_overuse(U, N) \;\&\; U \leq 0$

$\Rightarrow \forall t', N'>N \quad state_{UA}(M, t', output(UA)) \not\models round(N')$

U4. Announcement generation effectiveness

The Utility Agent satisfies *announcement generation effectiveness* if

for each initiated negotiation round, for each cutdown fraction
at least one reward value is announced

Formally:

$\forall M \in Traces(UA)\ \forall t, N$

$[\, state_{UA}(M, t, output(UA)) \models round(N)$

$\Rightarrow \exists t' \geq t\ \forall CD\ \exists R\ \ state_{UA}(M, t', output(UA)) \models announcement(CD, R, N)\ ]$

U5. Announcement uniqueness

The Utility Agent satisfies *announcement uniqueness* if

for each initiated negotiation round at most one announcement is generated

Formally:

$\forall M \in Traces(UA)\ \forall t, t', N\ \forall CD, R, R'$

$state_{UA}(M, t, output(UA)) \models announcement(CD, R, N)$

$\&\ \ state_{UA}(M, t', output(UA)) \models announcement(CD, R', N)$

$\Rightarrow \quad R = R'$

U6. Announcement generation groundedness

The Utility Agent satisfies *announcement generation groundedness* if

an announcement is only generated for initiated negotiation rounds

Formally:

$\forall M \in Traces(UA)\ \forall t, N\ \forall CD, R$

$state_{UA}(M, t, output(UA)) \models announcement(CD, R, N) \quad \Rightarrow \quad \exists t' \leq t \quad state_{UA}(M, t', output(UA)) \models round(N)$

U7. Monotonicity of announcement

The Utility Agent satisfies *monotonicity of announcement* if

for each announcement and each cut-down percentage the offered reward is at least
the reward for the same cut-down percentage offered in the previous announcements

Formally:

$\forall M \in Traces(UA)\ \forall t, t', N, N'\ \forall CD, R, R'$

$state_{UA}(M, t, output(UA)) \models announcement(CD, R, N)$

$\&\ \ state_{UA}(M, t', output(UA)) \models announcement(CD, R', N')$

& $N \leq N'$

$\Rightarrow$ $R \leq R'$

U8. Progress in announcement

The Utility Agent satisfies *progress in announcement* if

for at least one cut-down percentage the difference between the currently announced reward and the previously announced reward is at least the positive constant m (announce margin)

Formally:

$\forall M \in$ Traces(UA) $\forall t, t', N \exists$CD R, R'

$state_{UA}(M, t, output(UA)) \models announcement(CD, R, N)$

& $state_{UA}(M, t', output(UA)) \models announcement(CD, R', N+1)$

$\Rightarrow R + m \leq R'$

U9. Announcement rationality

The Utility Agent satisfies *announcement rationality* if

no announced reward is higher than the maximal reward

Formally:

$\forall M \in$ Traces(UA) $\forall t, N$ $\forall$CD, R

$state_{UA}(M, t, output(UA)) \models announcement(CD, R, N) \Rightarrow R \leq mr_{UA}(CD)$

U10. Transfer successfulness and groundedness properties within UA

Similar to the communication successfulness and groundedness properties at the level of the whole system (i.e., properties S5 and S6), also within the agents properties are needed that guarantee proper information transfer between the different components. These properties can be specified in a form as in S5 and S6. For example, proper transfer of Customer Agents bids from Agent Interaction Management to Determine Balance requires:

Transfer successfulness from AIM to DB:

$\forall M \in$ Traces(UA) $\forall$CA, t, CD, N

$state_{UA}(M, t, output(AIM)) \models cutdown\_from(CD, CA, N)$

$\Rightarrow \exists t' \geq t$ $state_{UA}(M, t', input(DB)) \models cutdown\_from(CD, CA, N)$

Transfer groundedness from AIM to DB:

$\forall M \in$ Traces(UA) $\forall$CA, t, CD, N

$state_{UA}(M, t, input(DB)) \models cutdown\_from(CD, CA, N)$

$\Rightarrow \exists t' \leq t$ $state_{UA}(M, t', output(AIM)) \models cutdown\_from(CD, CA, N)$

Similar transfer properties are used for all other information transfer between components within UA. For the proofs these properties are assumptions, on the basis of the design specification: the transfer properties are the expression of the semantics of the information links in a design.

U11. Environmental property for UA
This environmental property states

if the Utility Agent makes an annoncement,
then after some time for each Customer Agent a bid has been received

Formally:

$\forall M \in$ Traces(S) $\forall$ t , N, CD, R

[$state_{UA}(M , t, output(UA)) \models round(N)$

& $state_{UA}(M , t, output(UA)) \models announcement(CD, R, N)$]

$\Rightarrow$ $\forall CA \; \exists t' > t, CD'$ $state_{UA}(M , t', input(UA)) \models cutdown\_from(CD', CA, N)$

6.2.2 Properties of each Customer Agent

C0. Finite termination of negotiation by CA
A Customer Agent CA satisfies *finite termination of negotiation by CA* if

a round number exists such that CA does not negotiate after this round number

Formally:

$\forall M \in$ Traces(S) $\exists N \; \forall t, CD, \; N'>N$ $state_S(M , t, output(CA)) \not\models cutdown(CD, N')$

A successfulness property of a Customer Agent could be defined on the basis of some balance between discomfort and financial gains. These aspects were not included in the model and the analysis.

C1. Bid generation effectiveness
A Customer Agent CA satisfies *bid generation effectiveness* if

for each announced negotiation round at least one bid is generated
(possibly a bid for reduction zero)

Formally:

$\forall M \in$ Traces(CA) $\forall t, N$

$state_{CA}(M , t, input(CA)) \models round(N) \Rightarrow \exists CD, t' \geq t$ $state_{CA}(M , t', output(CA)) \models cutdown(CD, N)$

C2. Bid uniqueness
A Customer Agent CA satisfies *bid uniqueness* if

for each negotiation round at most one bid is generated

Formally:

$\forall M \in$ Traces(CA) $\forall t, t', N, CD, CD'$

$state_{CA}(M, t, output(CA)) \models cutdown(CD, N)$ & $state_{CA}(M, t', output(CA)) \models cutdown(CD', N)$
$\Rightarrow$ CD = CD'

C3. Bid generation groundedness

A Customer Agent CA satisfies *bid generation groundedness* if

a bid is only generated once a negotiation round is announced

Formally:

$\forall M \in$ Traces(CA) $\forall$ t, N, CD
$state_{CA}(M, t, output(CA)) \models cutdown(CD, N) \Rightarrow \exists\, t' \leq t \quad state_{CA}(M, t', input(CA)) \models round(N)$

C4. Bid rationality

A Customer Agent CA satisfies *bid rationality* if

for each bid the required reward for the offered cut-down is at most the reward announced in the same round, and the offered cut-down is the highest with this property

Formally:

$\forall M \in$ Traces(CA) $\forall$ t, t', N, CD, R
[[$state_{CA}(M, t, output(CA)) \models cutdown(CD, N)$ & $state_{CA}(M, t', input(CA)) \models announcement(CD, R, N)$]
$\Rightarrow$ $rr_{CA}(CD) \leq R$] &
$\forall$ t, t',t", N, CD, R, CD',R' [[$state_{CA}(M, t, output(CA)) \models cutdown(CD, N)$
& $state_{CA}(M, t', input(CA)) \models announcement(CD, R, N)$
& $state_{CA}(M, t", input(CA)) \models announcement(CD', R', N)$
& $rr_{CA}(CD') \leq R'$]
$\Rightarrow$ CD $\geq$ CD']

Note that the choice to take the highest cutdown of the acceptable cutdowns is a bit arbitrary. If the model of the Customer Agents is specialised to a more refined model incorporating more detail on a user profile model involving the balance between discomfort and financial gain, another, more sophisticated criterion for this decision could be used instead. However, the system was designed with focus more on the Utility Agent and the Customer Agents in the actual system make the decision as indicated in property C4; since the analysis concerns the system as designed, the above property has been specified as above to fit the system.

C5. Monotonicity of bids

A Customer Agent CA satisfies *monotonicity of bids* if

for each bid the offered cutdown value is at least as high (a cut-down percentage) as for the bids in the previous rounds

Formally:

$\forall M \in$ Traces(CA) $\forall$ t, t', N, N' $\forall$ CD, CD'

$state_S(M, t, output(CA)) \models cutdown(CD, N)$ & $state_S(M, t', output(CA)) \models cutdown(CD', N')$ & $N \leq N'$
$\Rightarrow$ $CD \leq CD'$

C6. Transfer successfulness and groundedness properties within CA
Similar to the agent UA, also for proper information transfer within CA transfer successfulness and groundedness properties are assumed; they have the same form as in U10.

6.2.3 Properties of the External World

The External World satisfies *information provision effectiveness* if it provides information about the predicted use of energy, the maximum energy level allocated to each Customer Agent, and the maximal overuse of the Utility Agent. The External World satisfies *static world* if the information provided by the external world does not change during a negotiation process.

6.3 Logical Relationships between Properties

To identify logical relationships between properties at different aggregation levels, the compositional structure of the system is followed. For the level of the whole system, system properties are proved from agent properties, which are defined at one process abstraction level lower.

6.3.1 Logical Relationships and Proofs for the System Properties

Property S4 is a domain-specific assumption on the system, which is used in the proofs of other properties. The top level properties S1, S2 and S3 can be proven from the agent properties in a rather trivial manner, since they happen to be conjunctions of agent properties. For example, by definition monotonicity of negotiation (S1) can be proven directly from the properties monotonicity of announcement (U7) and monotonicity of bids (C5) for all Customer Agents, since it is the conjunction of these properties. In a similar manner system property S2 (termination) can be proven directly from U0 and C0, and system property S3 (rationality) from U9 and C4. Properties S5 and S6 are generic assumptions for the system as a whole: within an implementation each arrow within the design has to be implemented in such a manner that transfer of information indeed takes place; the prototype implementation satisfies these requirements due to the fact that it was built using the DESIRE implementation generator which satisfies such requirements in a generic manner.

6.3.2 Logical Relationships and Proofs for Agent Properties

Agent properties can be logically related to (and proven from):

1. Properties of sub-components of the agent.
   In this case the proof can be made at one process abstraction level lower: purely within the agent. This is discussed for the Utility Agent in Section 7.
2. Other agent properties.
   An example proof of this type will be discussed below: for the Utility Agent property U0 (from U1, U3 and U6).
3. Agent properties relating to the agent's environment.
   An example proof of this type will be discussed below: for Utility Agent property U11 (from Customer Agent property C1 and S5)

Also combinations of types 1. or 3. with 2. are possible. An example of this is the proof of the Customer Agent property C0 (from C3 and U10) shown below.

*** Proof of U0 from U1, U3 and U6.**

As a first example, the termination property for the Utility Agent (U0)

$$\forall M \in \text{Traces}(S)\ \exists N\ \forall N'>N, t, CD, R \quad \text{state}_S(M, t, \text{output}(UA)) \not\models \text{announcement}(CD, R, N') \ \&\ \text{state}_S(M, t, \text{output}(UA)) \not\models \text{round}(N')$$

can be proven from the properties U1, U3, and U6. The proof runs as follows. Let a trace $M \in \text{Traces}(S)$ be given. From U1 it follows that for some $N, t,$ and $U \leq \text{max\_overuse}$ it holds

$$\text{state}_S(M, t, \text{output}(UA)) \models \text{predicted\_overuse}(U, N)$$

By property U3 it follows that

$$\forall t', N'>N \quad \text{state}_{UA}(M, t', \text{output}(UA)) \not\models \text{round}(N')$$

From this, using property U6 it follows:

$$\forall t', N'>N, R, CD \quad \text{state}_{UA}(M, t', \text{output}(UA)) \not\models \text{announcement}(CD, R, N')$$

This proves U0.

*** Proof of U11 from C1 and S5.**

Proofs of an environmental agent property takes place at the system level. As an example, the proof of U11 is shown. Based on the properties C1 and S5, the proof runs as follows. Let $M \in \text{Traces}(S)$ be given, and t, N, U, CD, R such that

$$\text{state}_{UA}(M, t, \text{output}(UA)) \models \text{round}(N)$$
$$\&\ \text{state}_{UA}(M, t, \text{output}(UA)) \models \text{announcement}(CD, R, N)$$
$$\&\ R < mr_{UA}(CD)$$

By communication successfulness (S5), for each CA a $t' \geq t$ exists such that

$$\text{state}_{CA}(M, t', \text{input}(CA)) \models \text{round}(N)$$

Using bid generation effectiveness (C1) it follows that for some $t'' \geq t'$ and CD it holds

$$\text{state}_{CA}(M, t'', \text{output}(CA)) \models \text{cutdown}(CD, N)$$

Again by communication successfulness (S5) for some $t''' \geq t''$ it holds.

$$\text{state}_{CA}(M, t''', \text{input}(UA)) \models \text{cutdown\_from}(CD, CA, N)$$

This proves the environmental property U11.

*** Proof of C0 from C3 and U10.**

The termination property of a Customer Agent depends on the Utility Agent, since the Customer Agents are reactive: the proof of C0 makes use of C3, the Utility Agent property U0, and the assumption that the communication between UA and CA functions properly.

What is to prove is that a Customer Agent CA satisfies finite termination of negotiation, i.e., a round number exists such that CA does not negotiate after this round number:

$$\forall M \in \text{Traces}(S)\ \exists N\ \forall t, CD, N'>N \quad \text{state}_S(M, t, \text{output}(CA)) \models \text{cutdown}(CD, N')$$

Let a trace $M \in \text{Traces}(S)$ and a Customer Agent CA be given. From Utility Agent property U0 it follows that an N exists such that:

$$\forall N'>N, t, CD, R \quad \text{state}_S(M, t, \text{output}(UA)) \models \text{announcement}(CD, R, N') \ \&\ \text{state}_S(M, t, \text{output}(UA)) \models \text{round}(N')$$

By communication groundedness (S6) it follows that

$$\forall N'>N, t \quad \text{state}_S(M, t, \text{input}(CA)) \models \text{round}(N')$$

By the bid generation groundedness property (C3), a bid is only generated by CA if a negotiation round was received; therefore:

$$\forall t, N'>N, CD \quad \text{state}_{CA}(M, t, \text{output}(CA)) \models \text{cutdown}(CD, N')$$

This proves Customer Agent property C0.

7. Compositional Verification within the Utility Agent

To illustrate the next level in the compositional verification process, in this section it is discussed how properties of the Utility Agent can be logically related to more basic properties: properties of components within the Utility Agent. First some of the properties of the components Agent Interaction Management and Determine Balance are defined.

7.1 Properties of Components within UA

Properties are defined for the components Agent Interaction Management (AIM), Determine Balance (DB), Cooperation Management (CM), and Own Process Control (OPC) of the Utility Agent (see Figure 1).

7.1.1 Properties of AIM

The following two properties express that the component Agent Interaction Management (1) distributes the relevant information from incoming communication, and (2) generates outgoing communication if required.

AIM1. Cut-down provision effectiveness

The component Agent Interaction Management satisfies *cut-down provision effectiveness* if AIM is effective in the analysis of incoming communication:

the cut-down information received by AIM of the form received(cutdown_from(CD, CA, N)) is interpreted and translated into cut-down information required by other components of the form offered_bid(cutdown(CD, CA, N)) and made available in AIM's output interface

Formally:

$\forall M \in$ Traces(AIM) $\forall$ t, N, CD, CA

state$_S$(M , t, input(AIM)) |= received(cutdown_from(CD, CA, N))

$\Rightarrow \exists$ t'≥t state$_S$(M , t, output(AIM)) |= offered_bid(cutdown(CD,CA,N))

AIM2. Communication generation effectiveness

The component Agent Interaction Management satisfies *communication generation effectiveness* if AIM is effective in generation of outgoing communication on the basis of the analysis of input information received from other components of the form next_communication(round(N)), next_communication(announcement(CD, R, N)) which is made available in statements of the form own_communication(round(N)), and own_communication(announcement(CD, R, N)):

$\forall M \in$ Traces(AIM) $\forall$ t, N, CD

state$_{AIM}$(M , t, input(AIM)) |= next_communication(X)

$\Rightarrow \exists$ t'≥t state$_{AIM}$(M , t, output(AIM)) |= own_communication(X)

7.1.2 Properties of Determine Balance

The following two properties express that the component Determine Balance calculates predictions in a reasonable manner.

DB1. Overuse prediction generation effectiveness

The component Determine Balance satisfies *overuse prediction generation effectiveness* if

the predicted overuse is determined if and when normal capacity,
predicted use and cut-downs are known

Formally:

$\forall M \in$ Traces(DB) $\forall$ t , N, C

state$_{DB}$(M , t, input(DB)) |= predicted_use(U)

& state$_{DB}$(M , t, input(DB)) |= normal_capacity(C)

& $\forall$CA CD state$_{DB}$(M , t, input(DB)) |= cutdown_from(CD,CA,N)

& state$_{DB}$(M , t, output(DB)) |= round(N)

$\Rightarrow \exists$U', t' ≥ t state$_{DB}$(M , t', output(DB)) |= predicted_overuse(U',N)

Here normal_capacity(C) denotes that the use C is preferred by the utility company, and cutdown_from(CD,CA,N) denotes that CD is the cutdown fraction offered by CA in round N.

DB2. Overuse prediction monotonicity

The component Determine Balance satisfies *overuse prediction monotonicity* if the following holds:

if based on received cut-downs CD_{CA} for each Customer Agent CA, a predicted overuse U is generated by DB, and based on received cut-downs CD'_{CA} for each Customer Agent CA, a predicted overuse U' is generated by DB, then $CD_{CA} \leq CD'_{CA,}$ for all CA implies $U' \leq U$

Formally:

$\forall M \in$ Traces(DB) $\forall$ t , t', N, N', C, U0, U, U'

state$_{DB}$(M , t, input(DB)) |= predicted_use(U0)

& $\forall$CA [state$_{DB}$(M , t, input(DB)) |= cutdown_from($CD_{CA,}$ CA, N)

& state$_{DB}$(M , t', input(DB)) |= cutdown_from($CD'_{CA,}$ CA, N')

& $CD_{CA} \leq CD'_{CA}$]

& state$_{DB}$(M , t, output(DB)) |= predicted_overuse(U, N)

& state$_{DB}$(M , t', output(DB)) |= predicted_overuse(U', N')

$\Rightarrow$ $U' \leq U$

Note that in this property the monotonicity is not meant over time, but for the functional relation between input and output of DB.

DB3. Overuse prediction decrease effectiveness

The component Determine Balance satisfies *overuse prediction decrease effectiveness* if the following holds:

cut-down values exist such that, if the Utility Agent's component Determine Balance receives them, the predicted overuse will be at most 0.

Formally, a collection of numbers CD_{CA} for each Customer Agent CA *exists* such that:

$\forall M \in$ Traces(DB) $\forall$ t , N

$\forall$CA state$_{DB}$(M , t, input(DB)) |= cutdown_from(CD_{CA}, CA, N)

$\Rightarrow \exists$ t'≥t, U ≤ 0 state$_{DB}$(M , t', output(DB)) |= predicted_overuse(U, N)

Note that this locally defined property only refers to imaginary numbers CD_{CA}. A number CD_{CA} is not referring to an actual property of Customer Agent CA. In contrast to this, system level property S4 (see Section 6.1) expresses the actual (assumed) properties of Customer Agents.

7.1.3 Properties of Cooperation Management

Cooperation Management fulfills a number of properties, for example on properly generation announcements: announcement generation effectiveness, announcement uniqueness, and announcement generation groundedness. These are defined similarly to the corresponding properties of the Utility Agent. In this paper only the property that guarantees that new rounds are initiated is explicitly stated.

CM1. Round generation effectiveness

The component Determine Balance satisfies *round generation effectiveness* if

CM determines the value of the next round and makes this information available to other components in its output interface:

Formally:

$\forall M \in$ Traces(CM) $\forall$ t , N

$$\text{state}_{CM}(M, t, \text{input(CM)}) \models \text{round(N)} \Rightarrow \exists\, t' \geq t\ \text{state}_{CM}(M, t', \text{output(CM)}) \models \text{round(N+1)}$$

7.1.4 Properties of Own Process Control

One of the properties of the component Own Process Control guarantees that decisions about continuation of a negotiation process are made:

OPC1. New announce decision effectiveness

This property expresses:

If the predicted overuse is still more than the maximum overuse, then a new announcement is warranted

Formally:

$\forall M \in$ Traces(OPC) $\forall$ t , N, U

$\text{state}_{OPC}(M, t, \text{input(OPC)}) \models \text{current\_negotiation\_state(predicted\_overuse(U, N))}$

& $\text{state}_{OPC}(M, t, \text{input(OPC)}) \models \text{current\_negotiation\_state(round(N))}$

& $U > \text{max\_overuse}$

$\Rightarrow \exists\, t' \geq t\ \text{state}_{OPC}(M, t', \text{output(OPC)}) \models \text{new\_announce}$

Here current_negotiation_state(predicted_overuse(U, N)) denotes that in the current state of the negotiation process the predicted overuse is U, and current_negotiation_state(round(N)) that round N is active. Moreover, new_announce denotes that a decision is made to make a next announcement.

7.2 Logical Relationships and Proofs within the Utility Agent

As an example, to prove the UA property U2 (*negotiation round generation effectiveness),* formalized by

$\forall M \in$ Traces(S) $\forall$ t , N, U, CD, R

$[\ \text{state}_{UA}(M, t, \text{output(UA)}) \models \text{round(N)}$

& $\text{state}_{UA}(M, t, \text{output(UA)}) \models \text{predicted\_overuse(U, N)}$

& $U > \text{max\_overuse}$

& $\text{state}_{UA}(M, t, \text{output(UA)}) \models \text{announcement(CD, R, N)}$

& $R < mr_{UA}(CD)\ ]$

$\Rightarrow \exists t' \geq t\ \text{state}_{UA}(M, t', \text{output(UA)}) \models \text{round(N+1)},$

a number of properties of sub-components, are of importance, and also the interaction between the components through the information links (the arrows in Figure 1) should function properly (transfer successfulness and groundedness property U10). The proof of property U2 uses properties

U10, U11, DB1, OPC1, CM1, and AIM2. The round number itself is determined by CM; to guarantee this, CM needs to satisfy the property of *round generation effectiveness* (CM1). This round value is transferred to the component AIM. The component AIM must fulfil the property of *communication generation effectiveness* (AIM2) to guarantee this value to be placed in the Utility Agent's output interface. Activation of the link to the Utility Agent's output interface depends on whether the component OPC derives the need for a new announcement. To guarantee this, the property *new announce decision effectiveness* (OPC1), is needed. Properties DB1 and U11 are needed to guarantee provision of input to CM.

Based on the properties mentioned, a sketch of the proof is as follows. Let $\mathbf{M} \in$ Traces(S) be given, and t , N, U, CD, R such that

$$\begin{aligned}
&\text{state}_{UA}(\mathbf{M}, t, \text{output(UA)}) \models \text{round(N)} \\
&\& \ \text{state}_{UA}(\mathbf{M}, t, \text{output(UA)}) \models \text{predicted_overuse(U, N)} \\
&\& \ U > \text{max_overuse} \\
&\& \ \text{state}_{UA}(\mathbf{M}, t, \text{output(UA)}) \models \text{announcement(CD, R, N)} \\
&\& \ R < mr_{UA}(CD)
\end{aligned}$$

By the environmental property U11, for each CA a $t' \geq t$ and CD exist such that

$$\text{state}_{CA}(\mathbf{M}, t', \text{input(UA)}) \models \text{cutdown\_from(CD, CA, N)}$$

Within the Utility Agent transfer successfulness and groundedness properties (U10) then guarantee that some time later the bid information is available in the input of its component DB (and is persisting there). In a similar manner it is proven that the predicted use information, round information and normal capacity occurs at the input of DB, i.e., for some $t'' \geq t$ it holds

$$\begin{aligned}
&\text{state}_{DB}(\mathbf{M}, t'', \text{input(DB)}) \models \text{predicted_use(U)} \\
&\& \ \text{state}_{DB}(\mathbf{M}, t'', \text{input(DB)}) \models \text{normal_capacity(C)} \\
&\& \ \forall CA\ CD\ \text{state}_{DB}(\mathbf{M}, t'', \text{input(DB)}) \models \text{cutdown_from(CD,CA,N)} \\
&\& \ \text{state}_{DB}(\mathbf{M}, t'', \text{output(DB)}) \models \text{round(N)}
\end{aligned}$$

Using property DB1 for this component then the current predicted overuse is derived:

$$\exists U', t''' \geq t\ \text{state}_{DB}(\mathbf{M}, t''', \text{output(DB)}) \models \text{predicted\_overuse(U',N)}$$

It can be assumed that the overuse for this round is above max_overuse (otherwise the conditions for U2 are not satisfied). Transfer to the component OPC (U10) and the property new announce decision effectiveness (OPC1), formalized by

$$\begin{aligned}
&\forall \mathbf{M} \in \text{Traces(OPC)}\ \forall t, N, U \\
&\quad \text{state}_{OPC}(\mathbf{M}, t, \text{input(OPC)}) \models \text{current_negotiation_state(predicted_overuse(U, N))} \\
&\quad \& \ \text{state}_{OPC}(\mathbf{M}, t, \text{input(OPC)}) \models \text{current_negotiation_state(round(N))} \\
&\quad \& \ U > \text{max_overuse} \\
&\quad \Rightarrow \ \exists\, t' \geq t, \text{state}_{OPC}(\mathbf{M}, t', \text{output(OPC)}) \models \text{new_announce},
\end{aligned}$$

then imply that this component will derive the atom new_announce. Given property CM1, formalized by

$$\forall \mathbf{M} \in \text{Traces(CM)}\ \forall t, N$$
$$\text{state}_{CM}(\mathbf{M}, t, \text{input(CM)}) \models \text{round(N)} \Rightarrow \exists t' \geq t\ \text{state}_{CM}(\mathbf{M}, t', \text{output(DB)}) \models \text{round(N+1)},$$

this component will derive a new round. Given property AIM2, formalized by

$$\forall \mathbf{M} \in \text{Traces(AIM)}\ \forall t, N, CD$$
$$\text{state}_{AIM}(\mathbf{M}, t, \text{input(AIM)}) \models \text{next\_communication(X)}$$
$$\Rightarrow \exists t' \geq t\ \text{state}_{AIM}(\mathbf{M}, t, \text{output(AIM)}) \models \text{own\_communication(X)},$$

the new round information own_communication(round(N+1)) will be available on the output interface of AIM; the transfer successfulness (U10) property related to the link outgoing communications is conditional in having new_announce (which is the case: from OPC1). Transfer of the desired result is therefore guaranteed: round(N+1) will occur at the output of UA. This shows how Utility Agent property U2 is proven.

8 Verification of Properties of Primitive Components

In Sections 6 and 7 compositional verification of the multi-agent system was described, providing logical relationships between properties at the different aggregation levels: for the system S as a whole, properties of the agents, and, the most basic properties: the properties of subcomponents of agents. Proofs of these logical relationships derive that the higher-level properties hold if the more basic properties of the subcomponents within the agents hold. If these components are considered primitive, i.e., not composed of other components, then their functionality can be described by a functional input-output relation (e.g., specified by a domain-specific knowledge base). For these primitive it has to be verified whether they satisfy the required properties. The primitive components specified by a knowledge-base can be verified – for not too large knowledge bases - making use of the more standard methods described in [25], [21]. This will be illustrated for the one of the components of the agent UA.

As an example, the component Determine Balance should satisfy the dynamic property ‘overuse prediction generation effectiveness’ (DB1) which expresses that if this component has received appropriate inputs for round N, then it will generate as an output an atom predicted_overuse(U', N) for some U':

$$\forall \mathbf{M} \in \text{Traces(DB)}\ \forall t, N, C$$
$$\text{state}_{DB}(\mathbf{M}, t, \text{input(DB)}) \models \text{predicted\_use(U)}$$
$$\&\ \ \text{state}_{DB}(\mathbf{M}, t, \text{input(DB)}) \models \text{normal\_capacity(C)}$$
$$\&\ \ \forall CA\ CD\ \text{state}_{DB}(\mathbf{M}, t, \text{input(DB)}) \models \text{cutdown\_from(CD,CA,N)}$$
$$\&\ \ \text{state}_{DB}(\mathbf{M}, t, \text{output(DB)}) \models \text{round(N)}$$
$$\Rightarrow\ \exists U', t' \geq t\ \text{state}_{DB}(\mathbf{M}, t', \text{output(DB)}) \models\ \text{predicted\_overuse(U',N)}$$

This dynamic property can be reformulated into the following static constraint C for component DB's input-output functionality relation, where $\wedge$ and $\vee$ stand for taking conjunctions resp. disjunctions:

$$\bigwedge_N [\bigvee_U \text{predicted\_use}(U) \ \wedge \ \bigvee_C \text{normal\_capacity}(C) \wedge \text{round}(N) \wedge \ \bigwedge_{CA} \bigvee_{CD} \text{cutdown\_from}(CD, CA, N)$$

$$\rightarrow \ \bigvee_{U'} \text{predicted\_overuse}(U', N)]$$

In terms of [21], the component can safely play the proper role if it is sound and strongly complete with respect to the domain description W(DB) defined by the constraint expressed above (i.e., the set of complete models satisfying the constraint), i.e.,

$$W(DB) = \{ M \in CMod(DB) \mid M \models C \}$$

In a similar manner the other dynamic properties of agents' subcomponents can be related to constraints on input-output relations. This shows how within a compositional verification process, required (possibly dynamic) properties of primitive components can be formulated as static properties of their input-output relation; cf. [25], [21].

9. Discussion

Verification of multi-agent systems is hard due to the fact that required properties for a multi-agent system usually refer to multi-agent behaviour which has nontrivial dynamics. Organisational structure is often used to constrain these multi-agent behavioural dynamics. An example in the area of negotiation is the use of negotiation protocols. The purpose of these negotiation protocols is to entail a structured process that is manageable within analysis, design and execution of such a multi-agent system. In this paper a case study on a multi-agent system for one-to-many negotiation in the domain of load balancing of electricity use was reported that shows that indeed verification becomes feasible. Compositional verification was applied to (1) logically relate dynamic properties of the multi-agent system as a whole to dynamic properties of agents, and (2) logically relate dynamic properties of agents to properties of their subcomponents. Given that properties of these subcomponents can be verified by more standard methods, these logical relationships provide proofs of the dynamic properties of the multi-agent system as a whole.

The design method DESIRE is based on compositionality of processes and knowledge at different levels of abstraction within a system design. To formally analyse such systems, the compositional verification method used in this paper fits well in addition to DESIRE, but it can also be useful to any other component-based design approach. The compositional verification method actually can be applied to a broad class of multi-agent systems. Compositional verification for one process abstraction level deep is based on the following very general assumptions:

- a multi-agent system consists of a number of agents and external world components.
- agents and components have explicitly defined input and output interface languages; all other information is hidden; information exchange between components can only take place via the interfaces (*information hiding*).
- a formal description exists of the manner in which agents and world components are composed to form the whole multi-agent system (*composition relation*).
- the semantics of the system can be described by the evolution of states of the agents and components at the different levels of abstraction (*state-based semantics*).

This non-iterative form of compositional verification can be applied to many existing approaches, for example, to systems designed using Concurrent METATEM (cf. [14]). Compositional verification involving more abstraction levels assumes, in addition:

- some of the agents and components are composed of sub-components.
- a formal description exists of the manner in which agents or components are composed of sub-components (*composition relation*).
- information exchange between components is only possible between two components at the same or adjacent levels (*information hiding*).

Currently not many approaches to multi-agent system design exist that exploit iterative compositionality. One approach that does is the component-based design method DESIRE. The compositional verification method used in this paper fits well to DESIRE, but not exclusively.

Two main advantages of a component-based approach to modelling are the transparent structure of the design and support for reuse of components and generic models. The compositional verification method extends these main advantages to (1) a well-structured verification process, and (2) the reusability of proofs for properties of components that are reused. The first advantage entails that both conceptually and computationally the complexity of the verification process can be handled by compositionality at different levels of abstraction. The second advantage entails: if a modified component satisfies the same properties as the previous one, the proof of the properties at the higher levels of abstraction can be reused to show that the new system has the same properties as the original. This has high value for a library of reusable generic models and components. The verification of generic models forces one to find the assumptions under which the generic model is applicable in the considered domain. A library of reusable components may consist of both specifications of the components, and their design rationale. As part of the design rationale, at least the properties of the components and their logical relations can be documented.

Also due to the compositional nature of the verification method, a distributed approach to verification is facilitated. This implies that several persons can work on the verification of the same system at the same time, once the properties to be verified have been determined. Since the proof of properties of a composed component depends on the properties of its sub-components, it is only necessary to know or to agree on the properties of these sub-components.

To obtain a more sophisticated formalisation of behavioural properties, different variants of temporal logic (cf. [4]) can be used, depending on the type of properties to be expressed. For example, linear or branching time temporal logic are appropriate to specify various agent (system) behavioural properties. Examples of formal requirement specification languages based on such variants of temporal logic are described in [9], [10], [14], [22]. However, for adaptive agents, it might be necessary to specify adaptive properties such as ‘exercise improves skill’ for which we have to explicitly express a comparison between different histories. This requires a form of temporal logic language which is more expressive than those allowing to model at each time point only one history. Therefore, in the language used in this paper, explicit reference to traces is made. Moreover it is chosen as a first-order predicate logic, which is not the case for most temporal logics.

To come to clearer understanding of strengths and weaknesses of a compositional approach to verification of multi-agent systems it is important to address real-world problems. The load balancing problem of electricity use, as addressed in this paper, belongs to the class of real-world problems. This paper focuses on the formal analysis of a prototype system for one-to-many negotiation between a Utility Agent and a (in principle large) number of Customer Agents, using a (monotonic) negotiation strategy based on announcing reward tables. This system was developed in

co-operation with Swedish electricity industry. For the analysis of this application the relevant behavioural properties were identified at the level of the multi-agent system as a whole, at the level of individual agents, and at the level of components within an agent. In addition the logical relationships between these properties were identified. As the case study aimed at analysis of this given system, the properties were formulated in a manner closely related to the system, in particular in terms of the information types used in input and output interfaces of agents or components within the design as given. No attempt has been done to specify the properties in a more generic manner; this certainly could be accomplished as a further extension of the work that might be useful for a larger class of applications.

As an evaluation of this exploration, in the first place it turned out that the properties that are relevant in the power load negotiation context could be specified and proven in an adequate manner, using the verification approach put forward. It turned out that in this application the relevant behavioural properties at the level of the multi-agent system as a whole have a rather direct relationship to behavioural properties of individual agents. This may seem a remarkable outcome, since often it is thought that behavioural properties of a multi-agent system in general have a complex relationship to properties of individual agents: the usual picture is that complex multi-agent system properties emerge from simple agent properties in a nontrivial manner. This may certainly be true in some classes of applications of multi-agent systems, for example, meant to model self-organisation, but the study reported in this paper shows that this is not always the case in industrial applications of multi-agent systems. This is also confirmed by some other case studies, for example, in the area of co-operative information agents (cf. [19], [20]), and in the area of organisation models for banking (cf. [13]), where relevant behavioural properties for the overall system have been identified and related to behavioural properties of individual agents in a transparent manner. In the application addressed in the current paper, the more nontrivial proofs can be found for the logical relationships between behavioural properties at the level of individual agents and components within agents.

A future continuation of this work will address [11] the development of tools for verification. To support the handwork of verification it would be useful to have tools to assist in the creation of the proof. This will be done by formalising the proofs of a verification process in a suitable proof system. Moreover, if as part of the design process, in addition to a design specification, requirements have been (formally) specified as well at different levels of process abstraction, these can be used as a useful starting point for a verification process; they provide a detailed map for the verification process and thus reduce the complexity by eliminating the search space for the requirement formulations at different process abstraction levels. Integration of the requirements engineering process within the system design process is also being addressed, since this leads to system designs that are more appropriate for verification than arbitrary architectures.

References

[1] Abadi, M. and Lamport, L., Composing Specifications, ACM Transactions on Programming Languages and Systems, vol. 15, No. 1, 1993, p. 73-132.

[2] Akkermans, H., Ygge, F., and Gustavsson, R., HOMEBOTS: Intelligent Decentralized Services for Energy Management. In: Proceedings of the Fourth International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK'96, 1996.

[3] Benjamins, R., Fensel, D., and Straatman, R., Assumptions of problem-solving methods and their role in knowledge engineering. In: W. Wahlster (ed.), Proceedings of the 12th European Conference on AI, ECAI'96, John Wiley and Sons, 1996, pp. 408-412.

[4] Benthem, J.F.A.K. van, The Logic of Time: A Model-Theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse, Dordrecht: Reidel, 1983.

[5] Brazier, F.M.T., Cornelissen, F., Gustavsson, R., Jonker, C.M., Lindeberg, O., Polak, B., and Treur, J., Agents Negotiating for Load Balancing of Electricity Use. In: M.P. Papazoglou, M. Takizawa, B. Krämer, S. Chanson (eds.), Proceedings of the 18th International Conference on Distributed Computing Systems, ICDCS'98, IEEE Computer Society Press, 1998, pp. 622-629.

[6] Brazier, F.M.T., Jonker, C.M., and Treur, J., Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering*, vol. 41, 2002, pp. 1-28.

[7] Brazier, F.M.T., Jonker, C.M., and Treur, J., Dynamics and Control in Component-Based Agent Models. *International Journal of Intelligent Systems*. In press, 2002.

[8] Brazier, F.M.T., Treur, J., Wijngaards, N.J.E., and Willems, M., Temporal semantics of compositional task models and problem solving methods. Data and Knowledge Engineering, vol. 29(1), 1999, pp. 17-42.

[9] Darimont, R., and Lamsweerde, A. van, Formal Refinement Patterns for Goal-Driven Requirements Elaboration. Proc. of the Fourth ACM Symposium on the Foundation of Software Engineering (FSE4), 1996, pp. 179-190.

[10] Dubois, E., Du Bois, P., and Zeippen, J.M., A Formal Requirements Engineering Method for Real-Time, Concurrent, and Distributed Systems. In: Proceedings of the Real-Time Systems Conference, RTS'95, 1995.

[11] Engelfriet, J., Jonker, C.M., and Treur, J., In: J.P. Mueller, M.P. Singh, A.S. Rao (eds.), Intelligent Agents V, Proc. of the Fifth International Workshop on Agent Theories, Architectures and Languages, ATAL'98. Lecture Notes in AI, vol. 1555, Springer Verlag, 1999, pp. 177-194.

[12] Fensel, D., Schonegge, A., Groenboom, R., and Wielinga, B., Specification and verification of knowledge-based systems. In: B.R. Gaines, M.A. Musen (eds.), Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems workshop, KAW'96, Calgary: SRDG Publications, Department of Computer Science, University of Calgary, 1996, pp. 4/1-4/20.

[13] Ferber, J., Gutknecht, O., Jonker, C.M., Mueller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems. In: Proc. of the Fourth International Conference on Multi-Agent Systems, ICMAS 2000. IEEE Computer Society Press, 2000. Extended version in: Proc. of the ECAI 2000 Workshop on Modelling Artificial Societies and Hybrid Organizations, MASHO'00, 2000.

[14] Fisher, M., and Wooldridge, M., On the Formal Specification and Verification of Multi-Agent Systems. In: International Journal of Cooperative Information Systems, M. Huhns, M. Singh, (eds.), special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, vol. 6, 1997, pp. 67-94.

[15] Gero, J.S., and Sudweeks, F., (eds.), Artificial Intelligence in Design '98, Kluwer Academic Publishers, Dordrecht, 1998.

[16] Gustavsson, R., Requirements on Information Systems as Business Enablers. Invited paper. In Proceedings of DA/DSM Europe DistribuTECH'97, PennWell, 1997.

[17] Hooman, J., Compositional Verification of a Distributed Real-Time Arbitration Protocol. In: Real-Time Systems, vol. 6, 1997, pp. 173-206.

[18] Jennings N. R. (2000) On Agent-Based Software Engineering. Artificial Intelligence, vol.117 (2) 277-296.

[19] Jonker, C.M., Klusch, M., and Treur, J., Design of Collaborative Information Agents. In: M. Klusch, and L. Kerschberg (eds.), Cooperative Information Agents IV, Proceedings of the Fourth International Workshop on Cooperative Information Agents, CIA 2000. Lecture Notes in Artificial Intelligence, vol. 1860, Springer Verlag, 2000, pp. 262-283.

[20] Jonker, C.M. and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380. Extended version in: *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

[21] Leemans, N.E.M., Treur, J., and Willems, M., A Semantical Perspective on Verification of Knowledge. *Data and Knowledge Engineering*, vol. 40, 2002, pp. 33-70.

[22] Manna, Z., and Pnueli, A., Temporal Verification of Reactive Systems: Safety. Springer Verlag, 1995.

[23] Roever, W.P. de, Langmaack, H., Pnueli, A. (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998.

[24] Rosenschein, J.S., and Zlotkin, G., Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, MIT Press, 1994.

[25] Treur J., and Willems M., A logical foundation for verification. In: A.G. Cohn (ed.), Proc. of the 11th European Conference on Artificial Intelligence, ECAI'94. John Wiley & Sons, Chichester, 1994, pp. 745-749

[26] Wooldridge, M., Jennings, N.R., and Kinny D. (1999). A Methodology for Agent-Oriented Analysis and Design. Proc. 3rd Int. Conference on Autonomous Agents (Agents-99) Seattle, WA, 69-76. Extended version: M. Wooldridge, N. R. Jennings, and D. Kinny (2000) The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems , vol. 3 (3) 285-312.