Knowl Inf Syst (2008) 15:1–30
DOI 10.1007/s10115-006-0049-z

Knowledge and Information Systems

REGULAR PAPER

Slobodan Ribarić · Tomislav Hrkać

TeMAS–a multi-agent system for temporally rich domains

Received: 8 November 2005 / Revised: 11 April 2006 / Accepted: 9 August 2006 /
Published online: 12 October 2006

Abstract In this paper, we present the model and simulator of a multi-agent system (MAS) for temporally rich domains. The theoretical foundations of the model include a knowledge representation scheme based on an original modification of Petri nets, called Petri nets with time tokens (PNTTs), as well as temporal reasoning based on the extension of Allen's temporal logic. The proposed MAS, called TeMAS, has a hierarchical structure, consisting of different levels, where each level contains clusters of agents. A paradigm of hierarchically organized blackboards is used for the communication among agents, clusters, as well as levels. We describe an object-oriented implementation of a program simulator of TeMAS and give an example of the use of the simulator for interpretation of events in a dynamic scene.

Keywords Multi-agent system · Petri nets · Knowledge representation · Temporal reasoning · Simulation

1 Introduction

Multi-agent systems (MASs) are becoming an increasingly important area in many fields of computer science, for example, robotics [29], image interpretation and computer vision [27, 41, 50], artificial life [30, 46], web applications [21] and many business and industrial applications [1, 16, 48], as well as in many other fields of artificial intelligence [47].

In general, a MAS can be defined as a collection of intelligent agents. According to Ferber [15], an intelligent agent, as a basic building block of a MAS, can be defined as a physical or virtual entity that (i) is capable of acting in an environment; (ii) can communicate directly with other agents; (iii) is driven by a

S. Ribarić (✉) · T. Hrkać
Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia
E-mail: slobodan.ribaric@fer.hr

set of tendencies; (iv) possesses resources of its own; (v) is capable of perceiving its environment; (vi) has only a partial representation of this environment; (vii) possesses skills and can offer services; and (viii) might be able to reproduce itself.

There are different definitions of an intelligent agent that is situated in some environment and that is capable of flexible autonomous action in this environment in order to meet its design objectives, where flexibility means reactivity, pro-activity, and social ability [54, 57].

Formally, a MAS was defined as n-tuple [15]:

$$MAS = (E, O, A, R, Op, LoU),$$

where E is the environment, i.e., a space that generally has a volume; O is a set of movable or stationary objects situated in E; A is a set of agents that can be considered as specific objects ($A \subseteq O$) representing active entities in the system; R is a set of relations among the objects and agents; Op is a set of the operations of agents, such as perceiving, transforming, and manipulating objects; and LoU is a set of so-called laws of the universe that are common for E. This paper proposes a model of a MAS called TeMAS (temporal MAS). This model is primarily intended to be a building block used in dynamic computer-vision systems that are the central component of intelligent systems performing sophisticated operations such as motion tracking, reasoning, and interpreting the behavior of several moving objects allocated in the scene. The above also includes event prediction in temporally rich domains, where temporally rich domains, according to the definition given by Pellavin and Allen [38], can be understood as domains that include concurrent actions that take time, the simultaneous occurrence of many actions, and domains with external events or natural forces.

The paper describes the three main aspects of a MAS which is adapted to computer-vision systems: first, the theoretical foundation of the reasoning agent based on an original high-level Petri-net model; second, the architecture of the hierarchically organized MAS with a paradigm of hierarchically organized blackboards used for the communication among agents; and finally, the object-oriented MAS simulator based on the two aforementioned aspects.

2 Related work

In the past three decades, different formal models have been proposed in an attempt to provide an effective solution to the problem of crisp or fuzzy temporal knowledge representation and reasoning: situation calculus [31], time specialist [22], interval-based temporal logic [2], formal logic of plans [38], time maps [12], metric constraints [13], timed, time and temporal Petri nets [9, 20, 32, 33, 49], fuzzy time Petri nets [36], and a point-interval logic [59]. In the paper [3], a temporal argumentation system is proposed. The authors give an alternative temporal logic allowing interval as well as instant-based temporal references, which improves in several aspects the widely used Allen's logic of intervals. In the paper [26], the elements of the spatio-temporal knowledge representation for dynamic environments are presented, where an event is defined as a finite set of objects that move in a given time along specified routes. The pattern is a set of "similar" events. Similarity based on the mathematical notion of similitude is used. In the

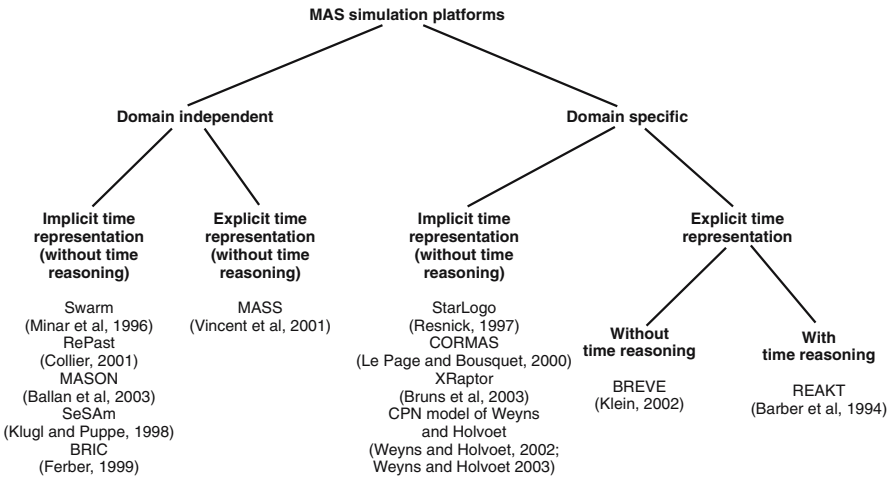

Fig. 1 Taxonomy of MAS simulation platforms

paper [58], the temporal index is adopted to realize the logical unified temporal data view and the seamless conjunction of non-temporal data and temporal data, and the unified temporal information model of temporal knowledge and temporal data has been established with temporal attribute.

The logical step during the development of the complex multi-agent-based system is its simulation on one of the simulation platforms [28, 51]. In Fig. 1, we propose the taxonomy of MAS simulation platforms and the corresponding MAS models related to the wider focus of our research. The taxonomy is based on the following characteristics: domain dependency, time representation, and the ability of time reasoning.

The existing MAS simulation platforms can be divided into domain-independent and domain-dependent platforms. One aspect of a MAS simulator is the way that time is represented. The existing domain-independent MAS simulators generally deal with time in an implicit way: agents are first modeled as separate entities with appropriate abilities, and then integrated into a MAS using a particular activation regime or scheduling algorithm [17]. This scheduling algorithm is responsible for the temporal order of the agent's actions, but it does not model the duration of the agent's actions and does not enable time reasoning. For example, in the Swarm simulation platform [35] the user defines how the time is simulated by writing a schedule of activities. The schedule contains a set of actions in a specified order. Individual actions take place at some specific time points, and time advances only by events scheduled at successive times. A very similar situation occurs for the time representation in the RePast simulator [11]. RePast behaves as a discrete-event simulator whose quantum unit of time is called *a tick*. Each event is scheduled for execution in a corresponding tick, so that a relative ordering of the events is enabled. In the MASON simulation platform [4], time is also represented implicitly by means of the so-called Discrete Event Scheduler, which permits agents to perform actions in time. In the SeSAm simulation platform [24], the activity of an agent at any moment of time is determined either according to a so-called skeletal plan associated with each agent, or by firing a rule

with a higher priority (in order to react to emergency situations). Ferber developed a BRIC language for modeling and simulating intelligent agents and MASs [15]. In BRIC, the agent's activities are modeled by means of Colored Petri Nets, but they are not time related. Therefore, the order of activities is determined by the Colored Petri Net execution rules.

Vincent et al. [52] developed a domain-independent multi-agent simulation platform called MASS, which includes an explicit model of time. In MASS, agents use a hierarchical task structure for the representation of their goals and capabilities. Goals are decomposed into sub-goals, and sub-goals are achieved by means of *methods*. One of the attributes associated with each method is its temporal duration.

Some domain-specific simulators also represent time in an implicit way. For example, in StarLogo [42], which is intended for the simulation of massively parallel systems, the activities of agents and their relative ordering are specified by means of a modification of the Logo programming language. The CORMAS simulator [25], intended for the modeling of renewable resources in natural and social dynamics, also employs a discrete-event simulation and a scheduler of the agent's activities. The XRaptor simulator [8] simulates the behavior of agents in continuous two- or three-dimensional worlds. The time line is divided by means of discrete time points, where differential equations are solved during each step of the simulation. Weyns and Holvoet proposed a Colored Petri Net based model for the space-situated agents where time is implicitly represented [55, 56].

Other domain-specific MAS simulators include an explicit time representation, but most of them do not have a temporal reasoning ability. The BREVE platform for the simulation of decentralized systems and artificial life [23] has a continuous time representation and enables the scheduling of events for execution at specified time points.

The REAKT multi-agent architecture [5] includes both an explicit time representation and a temporal reasoning ability. Temporal reasoning allows a selection of the agent's actions based on temporal considerations. The REAKT architecture is built around a temporal blackboard, which provides the storage and management of objects used by a set of agents. Each object instance is defined by its attributes or slots. There are two types of slots: static slots, the value of which does not change over time; and temporal slots, the value of which changes over time. Temporal constraints among data are kept in a temporal graph named the Time Map, which is managed by a module called the Time Map Manager. A point-based temporal model is used, and time intervals are defined as pairs of their extreme time points: (begin, end). Thus, the time corresponding to an event can be exactly known (both the time points of the beginning and end are known, which is usually the case with past events), partially known (the time of the beginning is known, but the end time is undefined; this is usually the case with current events), known with imprecision (an event starts or finishes at some point within an interval) and temporally dependent (the starting or ending time of an event depends on other temporal facts). The temporal reasoning in REAKT is performed by means of rule-based agents.

According to the proposed taxonomy (Fig. 1), our MAS model and the simulation platform can be classified as domain specific, with explicit time representation and reasoning.

3 Theory outline for the proposed MAS

In this section, we briefly describe the theoretical principles that are necessary for an understanding of the proposed MAS model. The theory includes a description of high-level Petri nets, called Petri nets with time tokens (PNTTs) [43], as well as some extensions of the original Allen's time interval logic, by means of which the relations between the time point and time interval, as well as the relations between two time points can be described.

3.1 Petri nets with time tokens

3.1.1 Definition of the PNTTs

PNTTs are high-level Petri nets based on the p-timed net model (which associates with each place p the time corresponding to the duration of an action or state) and a new concept of time tokens. A time token, like a token in the Colored Petri Nets [18, 19], has "individuality," i.e., it carries inherent information about the current time of its detainment at the visiting places during the execution of the net.

Formally, the PNTT is defined as 9-tuple:

$$PNTT = (P, T, I, O, \tau, M, \upsilon, \delta, \Omega).$$

In the definition given earlier, P, T, I, and O are the components of ordinary Petri nets (PNs), defined as follows [39, 40]: P is a finite set of places $P = \{p_1, p_2, \ldots, p_n\}, n \geq 0$, T is a finite set of transitions $T = \{t_1, t_2, \ldots, t_m\}$, $m \geq 0$, I is an input function $I : T \rightarrow P^{\infty}$, a mapping from transitions into bags of places, O is an output function $O : T \rightarrow P^{\infty}$, a mapping from transitions into bags of places, $P \cap T = \emptyset$.

A function $\tau : P \rightarrow (Q^* \cup \infty)$ is a mapping from a set of places P to a set of time delays (Q^* is the set of non-negative rational numbers). A $\tau(p_i) \in (Q^* \cup \infty)$ corresponds to the time duration of some action or state associated with the place p_i. The value $\tau(p_i)$ can be considered as a time delay of tokens at the place p_i. All $\tau(p_i), i = 1, 2, \ldots, n$, where n is the cardinality of a set P, form a set of time delays $TD = \{\tau(p_1), \tau(p_2), \ldots, \tau(p_n)\}$.

A set $M = \{m_1^1, m_1^2, \ldots, m_1^{j-1}, m_1^j, m_2^1, m_2^2, m_2^3, \ldots, m_2^k, \ldots, m_r^1, \ldots, m_r^p\}$, $1 \leq r < \infty$, is a set of time tokens. A time token m_i^{k-1} is an ancestor of a time token m_i^k (see Sect. 3.1.3).

The function $\upsilon : M \rightarrow \prod_{1 \leq i < \infty}(P \times TD)_i$, where $\prod$ represents a Cartesian product, where P is a set of places, TD is a set of time delays, and $\times$ denotes the Cartesian product, is a mapping called a time track. A $\upsilon(m_i^k), m_i^k \in M$, determines the sequence of places and time delays that is assigned to the time token m_i^k as a consequence of the execution of the PNTT. It determines the time track for m_i^k that is formed on the basis of visiting places and the time delays of all time tokens that are ancestors of m_i^k.

The function $\delta : M \rightarrow (Q^* \cup \infty)$ is a mapping called a time accumulation. It determines the current amount (current accumulated time) of detaining time for the token $m_i^k \in M$ that is obtained during the execution of the PNTT. An initial

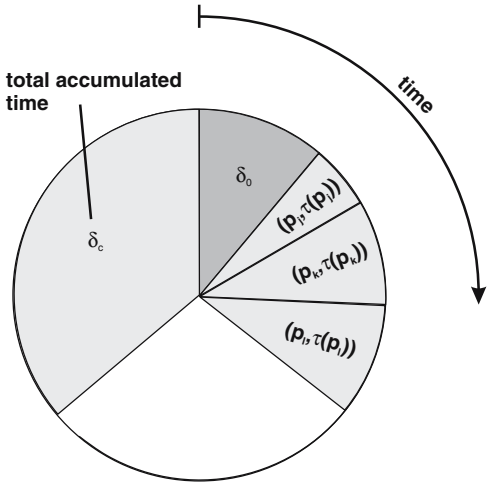

Fig. 2 The graphical representation of a time-token structure

time accumulation δ_0 can be assigned to a time token that is initially distributed in the PNTT. It can be interpreted as an initial extension of the time interval of the corresponding place corresponding to this time token.

By means of the functions υ and δ, the structure of a time token is defined. It carries information about the initial time accumulation, about the time of detainment of its ancestor time tokens at the visiting places and about the current accumulated time. This information (the time track and the current accumulated time) will be used for temporal reasoning. In general, the structure of a time token can be represented by the n-tuple $(\delta_0, ((p_j, \tau(p_j)), (p_k, \tau(p_k)), \ldots, (p_l, \tau(p_l))), \delta_c)$, where δ_c is the current accumulated time. The structure of a time token is graphically depicted in Fig. 2.

An injective function $\Omega : P \rightarrow \wp(M)$, with the restriction that $\Omega(p_i) \cap \Omega(p_j) = \emptyset$, for $i \neq j$, is called the marking of the PNTT. The $\wp(M)$ denotes the power set of M. With Ω_0 we denote the initial marking, i.e., the initial distribution of time tokens in the places of the PNTT.

3.1.2 PNTT graph

The PNTT can be represented by a bipartite directed multigraph. The circles represent places, while the bars represent transitions. The directed arcs connecting the places and transitions are defined by means of an input function I, while the arcs directed from the transitions to places are defined by an output function O. Multiple input places and multiple output places are represented by multiple arcs. The time tokens are represented by dots (•) in the places. Due to the "individuality" of tokens, every dot is labeled with $m_i^k \in M; i = 1, 2, \ldots, r; k = 1, 2, \ldots$

3.1.3 Execution rules for PNTT

Generally, tokens give dynamical properties to the PN, and they are used to define the execution of a PN: by firing an enabled transition t_j, tokens are removed from

its input places (elements of $I(t_j)$). Simultaneously, new tokens are created and distributed to its output places (elements of $O(t_j)$) [39]. In the PNTT, a transition is enabled if each of its input places has at least as many tokens in it as arcs from the place to the transition and if the time of detainment of these tokens in the places has elapsed. Such tokens are called movable time tokens. The firing of an enabled transition in the PNTT is performed automatically and immediately after the transition is enabled. The number of tokens at the input and output places is changed in accordance with the basic definition for an original PN. The firing of the enabled transition in the PNTT removes time tokens from its input places and simultaneously generates time tokens–successors in its output places. Due to the individuality of time tokens, the time tokens at output places can be viewed as new tokens that have additional (new) information about the time of stay at the corresponding (input) places. Each time token carries a specific history of the execution of the PNTT, and its history depends on the time token–ancestor's path through the PNTT. If the number of movable time tokens at the input place p_i is larger than $\#(p_i, I(t_j))$ for the enabled transition t_j (where # denotes a number of appearances of p_i in the bag $I(t_j)$), then different selection strategies for choosing the time tokens–ancestors that will be involved in the firing process can be used. For example, (i) the random selection of the time tokens; (ii) the last-in-first-out (LIFO) strategy, based on the arrival of the tokens into the place; (iii) the first-in-first-out (FIFO) strategy; or (iv) the selection of the time tokens with the stormiest (time) history, i.e., the time tokens with the most complex structure. In our temporal knowledge representation scheme, we use strategy (iv) to specify the time tokens–ancestors because these time tokens contain the richest information needed for temporal reasoning, and they are used for the generation of time tokens–successors.

Example 1 Figure 3a shows the marked PNTT graph defined as: $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2\}$, $I(t_1) = \{p_3, p_4\}$, $O(t_1) = \{p_1\}$, $I(t_2) = \{p_1, p_2\}$, $O(t_2) = \{p_1, p_4\}$, $\tau(p_1) = \tau_1$, $\tau(p_2) = \tau_2$, $\tau(p_3) = \tau_3$, $\tau(p_4) = \tau_4$; $\tau_1 = \tau_2 > \tau_3 = \tau_4$, the initial marking $\Omega_0 = (\{m_1^1\}, \{m_2^1, m_3^1\}, \{m_4^1\}, \{\emptyset\})$. After the time of detainment of the tokens at places p_1 and p_2 has elapsed, the time tokens become movable, transition t_2 is enabled, and it fires automatically. The firing of transition t_2 results in a new marking, where the old time tokens are removed from its input places and their successors are created at its output places. The marking of the PNTT after the firing of t_2 and the new structure of time tokens–successors are shown in Fig. 3b and e, respectively. The new marking Ω is: $(\{m_1^2\}, \{m_2^1\}, \{m_4^1\}, \{m_1^{2\prime}\})$, where time tokens m_1^2 and $m_1^{2\prime}$ are successors of the time token m_1^1, and both of them have the same structure. Let us suppose that $\tau_3 = \tau_4 < \tau_1$. After the time τ_4 has elapsed, the time token $m_1^{2\prime}$ becomes movable (the time token m_4^1 has already been movable), the transition t_1 is enabled and fires automatically. The new marking is depicted in Fig. 3c. The corresponding structures of the time tokens are shown in Fig. 3f.

3.1.4 Formal analysis of the PNTT

A formal analysis of the PNTT can be achieved by means of modified, well-elaborated procedures for ordinary Petri nets [39, 40]. In general, analysis prob-

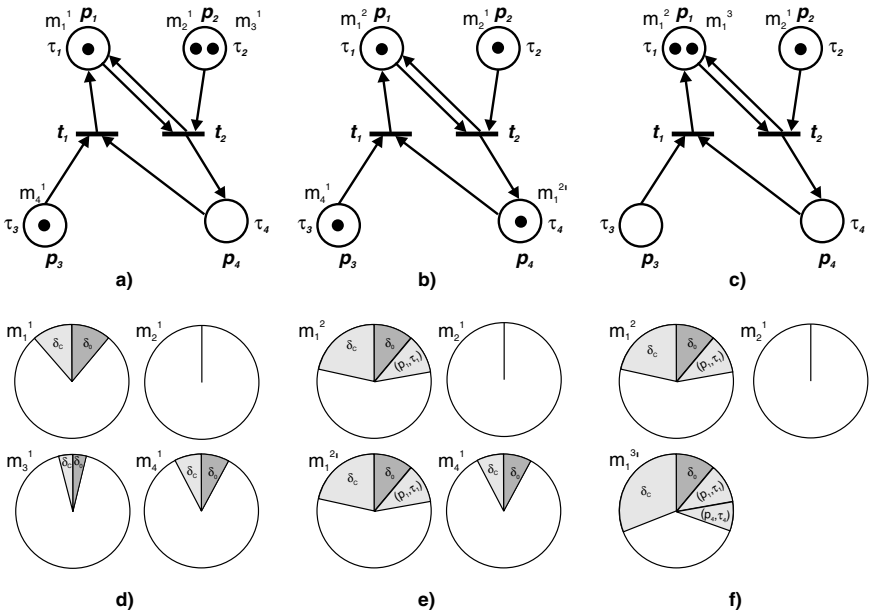

Fig. 3 **a** Initial marking of the PNTT. **b** Marking of the PNTT after the firing of t_2. **c** Marking of the PNTT after the firing of t_1. **d** The structure of time tokens before the firing of the transition t_2. **e** The structure of time tokens after the firing of t_2. **f** The structure of time tokens after the firing of t_1

lems for Petri nets, such as safeness, boundedness, conservation, liveness, reachability, etc., are based on a reachability tree analysis. Due to the specific application of our model, instead of the reachability tree with additional time information, we have proposed the firing diagram. The firing diagram is represented by a two-dimensional space: discrete-time and time tokens, in which the visited places and fired transitions are specified for each time token. In the firing diagram, the x-axis represents the discrete time, while the y-axis contains the time tokens. From the firing diagram, one can see the time detainment of time tokens, the sequence of firing-enabled transitions, as well as the reached states or the actions in the model, where the indexes for τ indicate the indexes of the places. The firing diagram depicted in Fig. 4 shows the execution of the PNTT from Example 1.

3.2 Temporal logic

Since our goal is the design of a MAS for reasoning in temporally rich domains, in this section we briefly describe the basic time primitives of the model. Two basic temporal primitives are used: a time point (also referred to as the date [14]) and a time interval [2]. Due to the specific application areas of our model, we have used the concepts of crisp time points and intervals, as well as crisp temporal relations.

The relations between two time intervals can be expressed by 13 Allen's temporal relations [2], which are shown in Table 1. By letting one of the time intervals degenerate to a time point, it is possible to define five relations between the time

Table 1 Allen's temporal relations

Relation	Symbol	Symbol for inverse	Pictorial example
X before Y	$<$	$>$	xxx yyy
X equal Y	$=$	$=$	xxx yyy
X meets Y	m	mi	xxxyyy
X overlaps Y	o	oi	xxx ..yyy
X during Y	d	di	xxx yyyyyy
X starts Y	s	si	xxx yyyyy
X finishes Y	f	fi	xxx yyyyy

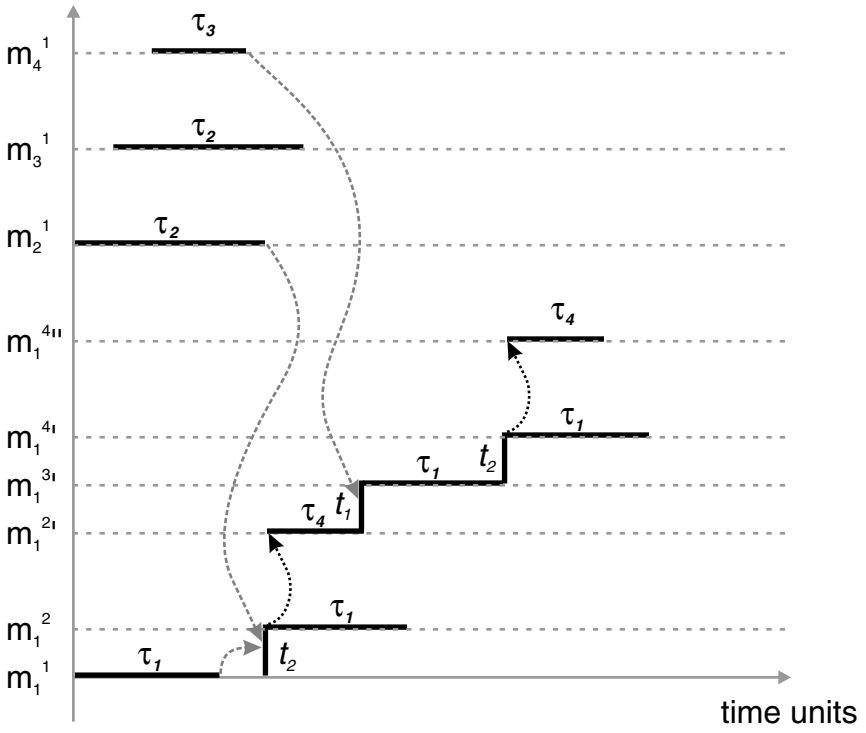

Fig. 4 Firing diagram for PNTT (Example 1)

point and the time interval (Table 2) [3]. By letting both of the time intervals degenerate to time points, three additional relations between the time points can be obtained (Table 3).

The proposed model can be extended by introducing concepts such as fuzzy time points, fuzzy intervals, and fuzzy relations [10, 14, 44].

Table 2 Temporal relations time point–time interval

Relation	Symbol	Symbol for inverse	Pictorial example
tp before X	$<$	$>$	• xxxx
tp during X	d	*not exists*	xx•xx
tp starts X	s	*not exists*	•xxxx
tp finishes X	f	*not exists*	xxxx•

Table 3 Temporal relations time point—time point

Relation	Symbol	Symbol for inverse	Pictorial example
tp1 before tp2	$<$	$>$	tp1• •tp2
tp1 equals tp2	$=$	$=$	tp1• tp2•
tp1 after tp2	$>$	$<$	tp2• •tp1

3.3 Temporal knowledge scheme

The temporal knowledge scheme KRPTT is defined as follows: $KRPTT = (PNTT, TLM, \alpha, \beta, F)$, where the PNTT is a Petri net with time tokens, while TLM is a module based on the extension of Allen's temporal logic. A function $\alpha : P \rightarrow (D \cup C_p)$ is a bijective function from a set of places P to a union of a set of actions and/or states D and a set of control states C_p. A function $\beta : T \rightarrow (\Sigma \cup C_t)$ is a mapping from a set of transitions T to a union of a set of events Σ and control events C_t. The meaning of C_p and C_t will be explained in the context of time-dependent event-driven reasoning. The functions α and β give the semantic interpretation of the model. The set F is a set of flags. In general, a flag $f_i \in F$ has a structure $f_i = (p_i, p_j, tr, p_l, p_k, \ldots, p_c)$, where p_i and p_j denote elements from P as places potentially with time tokens that have to be tested by the TLM, according to the temporal relation tr. The evaluation of the tr is based on information that is carried by the time tokens from the places p_i and p_j. The temporal relation tr is from a set of 13 possible Allen's time-interval relations (Table 1) defined in [2], five relations between time point and time interval (Table 2) and three relations between time point and time point (Table 3). In flags, tr can be compounded by using the logical connective *OR*. The $p_k, \ldots, p_c$ in the flag f_i specify the places (called control places) in which the TLM sets tokens depending on the result of the evaluation of tr. These tokens, called control tokens, are treated as time tokens with zero current accumulated time (time tokens without a time history).

A degenerative type of flag $f_{Gi} = (p_h, -, -, -)$ is used to denote the goal state of the system.

A temporal logic module (TLM) is a component of the KRPTT scheme that is capable of inferring about the temporal relations between two time intervals, between the time point and the time interval, and finally between two time points. The evaluation of tr is activated when one of the time tokens is available in the specified place and when the time of its stay in the place has elapsed, i.e., the token becomes movable (for temporal relations $<$ (before) and $>$ (after)) or when both time tokens are available and movable at the places specified by the flag. At that moment, the copies of the time tokens are transferred to the

TLM. The TLM is data driven, and the transfer of the copies of time tokens activates the evaluation of the temporal relation. Immediately after the evaluation, if the relation *tr* is satisfied, the control tokens are transferred to control places $p_k, \ldots, p_c$.

3.3.1 Reasoning

The time reasoning in the KRPTT, which allows an agent to infer about the temporal relations and to achieve time-dependent goals, is based on the dynamical properties that are manifested by the execution of the PNTT. The process of the reasoning in the proposed representation scheme is called the time-dependent event-driven reasoning. It includes not only the ability to reason about time relations between actions and/or states in the temporally rich domains but also the ability to determine the subsequent activities or states based on initial or current activities (states) and temporal information associated with each activity (state). The reasoning process can be described as follows. The input is the initial marking of the KRPTT representation of the world. It determines the current activities (states) and the temporal relations among the activities (states). As time proceeds, the enabled transitions are automatically fired and time tokens are distributed through the PNTT. Depending on the path through the PNTT and the time duration of the activities (states), each time token carries a history of the execution of the PNTT. The firing sequences are additionally controlled by the TLM in such a way that it tests and evaluates the temporal relations between activities (states), which correspond to the places specified by flags. According to the results of the temporal relationship evaluation, the control tokens are put at the control places (which correspond to the control states $c_p \in C_p$) specified by the flags. These tokens have an influence on the result of the firing sequence in the PNTT. The combination of the time tokens associated with activities (states) and control tokens, both present at the same time, can be interpreted as a time-dependent if–then rule implementation. If there are enough time and control tokens at the corresponding input places, the transition (corresponding to the control event $c_t \in C_t$) is fired, i.e., the time-dependent rule is activated and the action (or conclusion) is generated. It is obvious that the process described earlier is driven by time-dependent events.

If some places in the PNTT are denoted as goal states, the scheme can conclude if these time-dependent goals can be achieved and detect the sequence(s) of activities (states) that lead to these goals. By applying the functions α and β, the semantic interpretation of the sequences mentioned earlier can be obtained. By varying the initial marking of the scheme and by changing the time durations of some activities or states, the scheme can be used for planning in temporally rich domains.

4 TeMAS model

Since the proposed MAS model is primarily intended to be used in knowledge-based computer-vision systems and it is well known that computer-vision systems, as well as many other complex systems, are hierarchically organized [6, 37, 34],

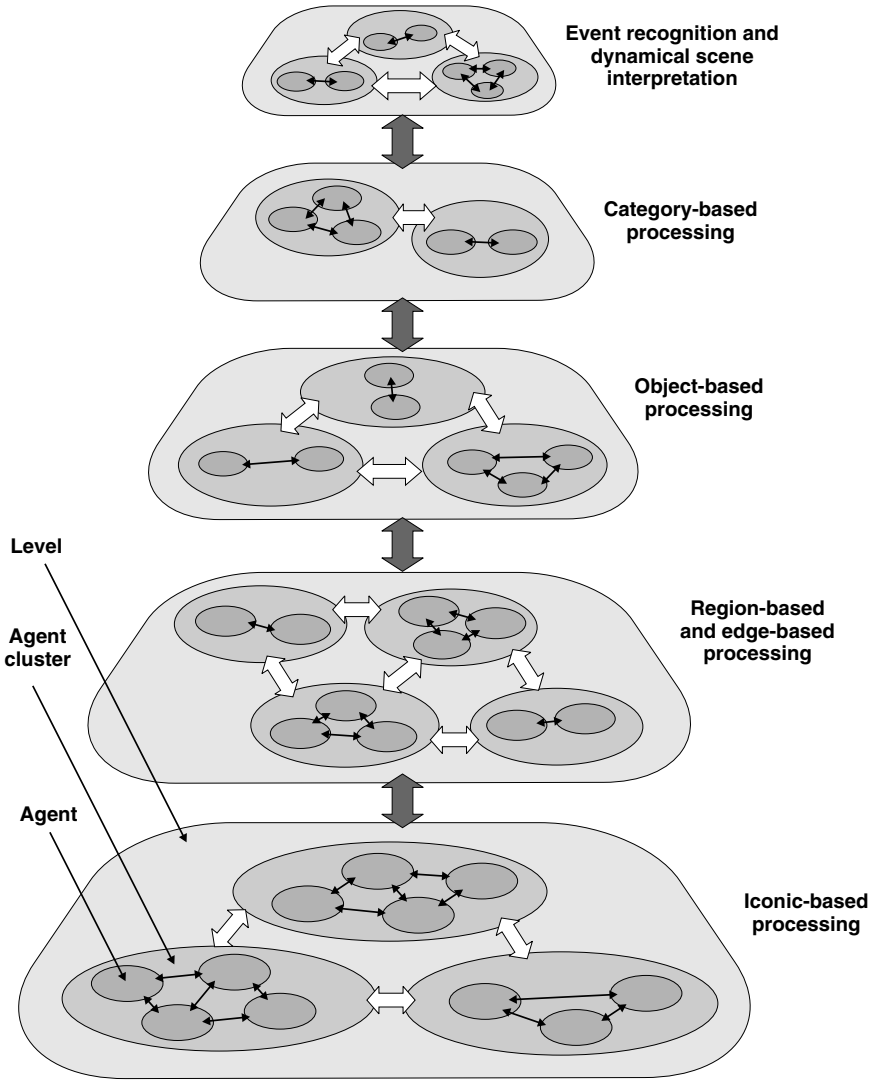

Fig. 5 The hierarchical structure of the MAS adapted to computer-vision activities

the organization of our MAS model is hierarchical. The MAS consists of several levels, which are composed of a number of clusters consisting of agents that perform identical or similar tasks. An example of the hierarchical structure of the MAS, adapted to computer-vision system activities, is depicted in Fig. 5.

The MAS models for the lower levels of computer-vision systems are based on behavioral agents [27, 53]. In this paper, we describe the MAS structure related to the higher levels of the computer-vision system (i.e., from category-based processing to event recognition and dynamic scene interpretation) called TeMAS.

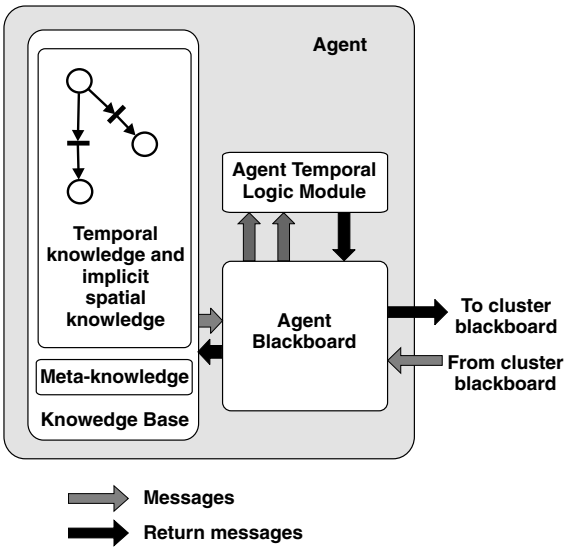

Fig. 6 The structure of an agent

4.1 Agent

An agent is the basic building block of the TeMAS model. Figure 6 depicts the structure of an agent.

As mentioned earlier, an agent generally possesses a partial representation of its environment. In the TeMAS model, the agent's representation of an environment consists of knowledge about temporal relations, implicit knowledge about spatial relations and a meta-knowledge. All these kinds of knowledge are contained in the agent's knowledge base. The knowledge about temporal relations as well as implicit knowledge about spatial relations in the agent's environment are represented by means of the KRPTT knowledge representation scheme. The agent's meta-knowledge (*aMK*) is implemented as a list of all the places of the PNTT that belong to the agent. As we shall see later, meta-knowledge has an important role in the communication among agents.

An agent also possesses a temporal reasoning ability, allowing it to infer about the relations between time primitives (time intervals and time points). This is the basis for achieving its time-dependent goals. In the proposed model, temporal reasoning is event driven. The basic agent component that is responsible for temporal reasoning is the agent's temporal logic module (*aTLM*). The aTLM tests the temporal relations between two time primitives that are contained in the structures (i.e., time track and current accumulation time) of the copies of time tokens in testing places denoted by flags. The agent's knowledge base and aTLM communicate by means of the agent's blackboard (*aBB*). When a time token arrives in the place of the PNTT denoted by a flag, the agent creates a message consisting of the structure of that time token and the structure of the corresponding flag, and places the message on the blackboard. The structure of the message is shown in Fig. 7a.

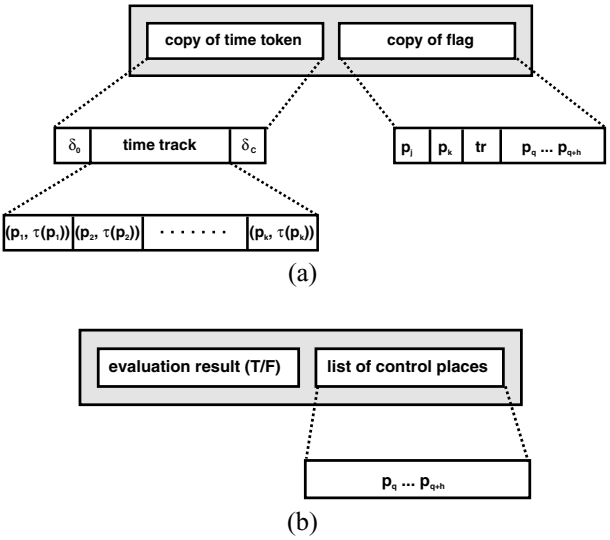

Fig. 7 **a** The structure of a message. **b** The structure of a return message

By inspecting the flag structure in the message and by comparing it with the agent's meta-knowledge (*aMK*), the agent determines whether another place denoted by the same flag is in the PNTT structure of the agent's local knowledge base. If it is so, the agent leaves the message on its own blackboard. When another time token arrives in another place denoted by the same flag, another message with the same structure is formed and placed on the agent's blackboard. When both messages (invoked by the same flag) are present on the agent's blackboard (except for the flags with temporal relations $<$ (before) and $>$ (after) where the arriving of only one message is enough), the agent's TLM is activated and it evaluates the specified temporal relation between the two time primitives specified in the messages. The aTLM returns a logical value: true or false. Based on the output of the aTLM, the return message is formed and placed on the agent's blackboard. The return message has a similar structure to that already described in the original message, but instead of a copy of the time token, it only specifies the logical value, i.e., the output of the aTLM. Instead of the complete flag structure, the other part of the return message contains only a list of the control places (Fig. 7b). When the return message arrives on the agent's blackboard, the agent inspects the logical value contained in the evaluation result (see Fig. 7b), and if this value is true the agent puts the control tokens into, by the message specified, control places, enabling certain control transitions of the PNTT in this way. The process mentioned earlier is related to the inside-agent communication.

4.2 Clusters and levels

Agents in the TeMAS model are organized into clusters. A cluster consists of a group of agents that perform identical or similar tasks. A cluster also has its

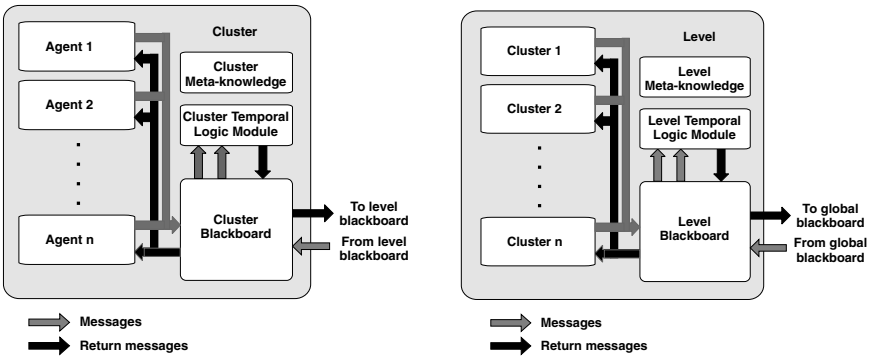

Fig. 8 **a** Cluster structure. **b** Level structure

own temporal logic module (*cTLM*), its blackboard (*cBB*), and its meta-knowledge (*cMK*). Figure 8a depicts the structure of a cluster.

One or more clusters at the same processing level are organized into a level of TeMAS. The structure of the level (Fig. 8b) is similar to the cluster structure: it consists of one or more agent clusters, a level temporal logic module (*lTLM*), a level blackboard (*lBB*), and the level's meta-knowledge (*lMK*).

All levels comprise the MAS. The MAS can be represented by a set of levels with a global temporal logic module (*gTLM*), a global blackboard (*gBB*), and a global meta-knowledge (*gMK*).

From the descriptions in Sects. 4.1 and 4.2, it can be seen that the TeMAS model has a four-level nested recursive structure (Figs. 6 and 8).

4.3 Communication among agents, clusters, and levels

The need for communication among different system components in TeMAS (i.e., agents, clusters, and levels) arises when a flag specifies that two time primitives, which are specified by two time tokens belonging to different agents, have to be compared. In such cases, the procedure described in Sect. 4.1. (inside-agent communication) cannot be performed because the two messages with corresponding time primitives (extracted from the time track) will not appear on the same agent's blackboard, and therefore neither agent's TLM will be able to perform the evaluation of the corresponding time relation.

The problem mentioned earlier can be solved by means of an evaluation of the time relation in the TLM belonging to the higher levels of the system hierarchy. In that case, the communication scenario is performed as follows. When a time token enters into a place marked by a flag, a corresponding message is placed on the agent's blackboard, as described in Sect. 4.1. Besides supporting a mechanism for the inside-agent communication, the agent's blackboard has an additional function. It can be considered as a starting point for a communication among other system components. Immediately after a message arrives onto the agent's blackboard, the agent, by means of its meta-knowledge, determines whether both testing places specified by the same flag belong to its own PNTT. If so, the agent

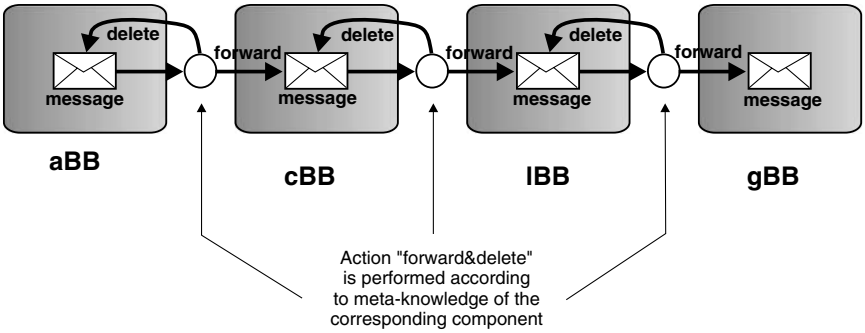

Fig. 9 The message routing

proceeds as we have described in Sect. 4.1. If the agent determines that one of the places denoted by the flag does not belong to its own PNTT, the agent performs the "forward and delete" action. It forwards the message to the blackboard of the next higher hierarchical component (i.e., a cluster blackboard) and deletes it from its own blackboard. When the message arrives to the blackboard of a higher-level component, a procedure similar to the one mentioned earlier is repeated. The component, by means of its meta-knowledge, checks whether both places belong to the PNTTs of the agents located inside the component. If so, the component waits for another message corresponding to the same flag, and when it arrives, the component's TLM is activated and the temporal relation is evaluated. If one of the specified places does not belong to the agents inside the component, it again performs a "forward and delete" action and forwards the message to the higher component's blackboard. For example, the message is forwarded from the cluster to the level. The described procedure of message routing is depicted in Fig. 9.

Based on the result of the evaluation of a temporal relation performed by an appropriate component's TLM (boolean value: true or false), a return message is formed and placed on the blackboard of the component that has performed the evaluation. Since, depending on the result of the evaluation, a control token has to be placed into the specified control place(s), the return message has to be routed to the corresponding agent(s). The routing procedure of the return message is based on the meta-knowledge of different system components and on the "backward and delete" action, similar to the already-described procedure of routing an original message, but in the opposite direction. After a return message arrives on the component's blackboard, the component consults its meta-knowledge and determines whether the specified control place is located in some of its sub-components or not. If so, the component returns the backward message to the blackboard of the corresponding sub-component and deletes it from its own blackboard. If the specified place is not located in any of the sub-components at the lower hierarchical levels, the component forwards the return message to the blackboard of the next hierarchically higher component. The described procedure repeats until the return message reaches the blackboard of the corresponding agent. The agent then puts or does not put a control token(s) in the specified control place(s), depending on the result of the evaluation, which is contained in the return message. The process of routing a return message is depicted in Fig. 10.

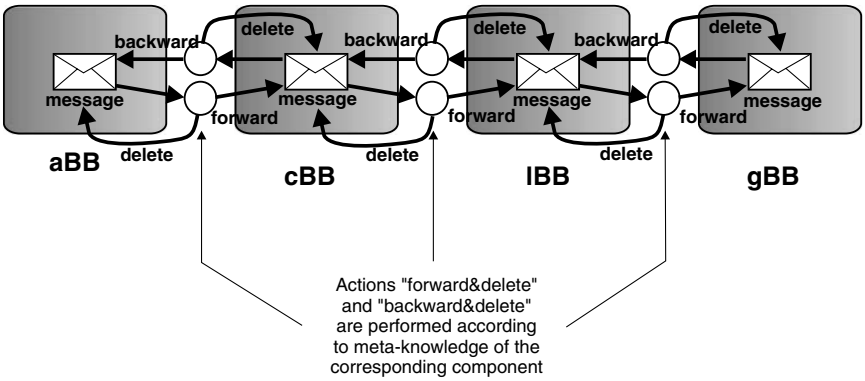

Fig. 10 The routing of a backward message

5 TeMAS simulator

5.1 Object-oriented implementation of the TeMAS simulator

Based on the proposed architecture of TeMAS, an object-oriented simulator was developed. In general, object-oriented design includes the following elements: abstraction into classes and objects, encapsulation, modularity, and inheritance with polymorphism [45].

The TeMAS structure is built from well-defined elements, with proper relationships; therefore, the abstraction of these elements into classes is straightforward. Four main classes are used to build this structure: *CMAS*, *CLevel*, *CCluster*, and *CAgent*, which represent different hierarchical components of the model (i.e., a MAS, a level, a cluster and an agent, respectively). Figure 11 depicts the simplified structure of the TeMAS simulator in UML class diagram notation [7]. The mentioned hierarchical components share a similar structure: each of them contains a blackboard, a TLM, a meta-knowledge, and either a set of lower-level entities (for example, MAS contains levels, a level contains clusters and a cluster contains agents) or a knowledge base in the form of a KRPTT (if the component is an agent).

For this reason, the inheritance mechanism was used for the implementation of the four classes mentioned earlier. All four of them inherit a base class called *CBaseAgent*.

The class *CBaseAgent* contains an instance of a class *CBlackboard*, an instance of a class *CTLM* and a list of integers that represent meta-knowledge, where each integer corresponds to an index of an appropriate place of the PNTT. The class *CBlackboard* represents a blackboard structure, and its main property is a list of messages. It also includes corresponding member functions for reading from and writing to a blackboard. The messages are implemented as instances of a class *CMessage*. The class *CMessage* reflects a message structure and is composed of a time token and a corresponding flag. The class *CTLM* represents a TLM, which is responsible for the evaluation of the temporal relations.

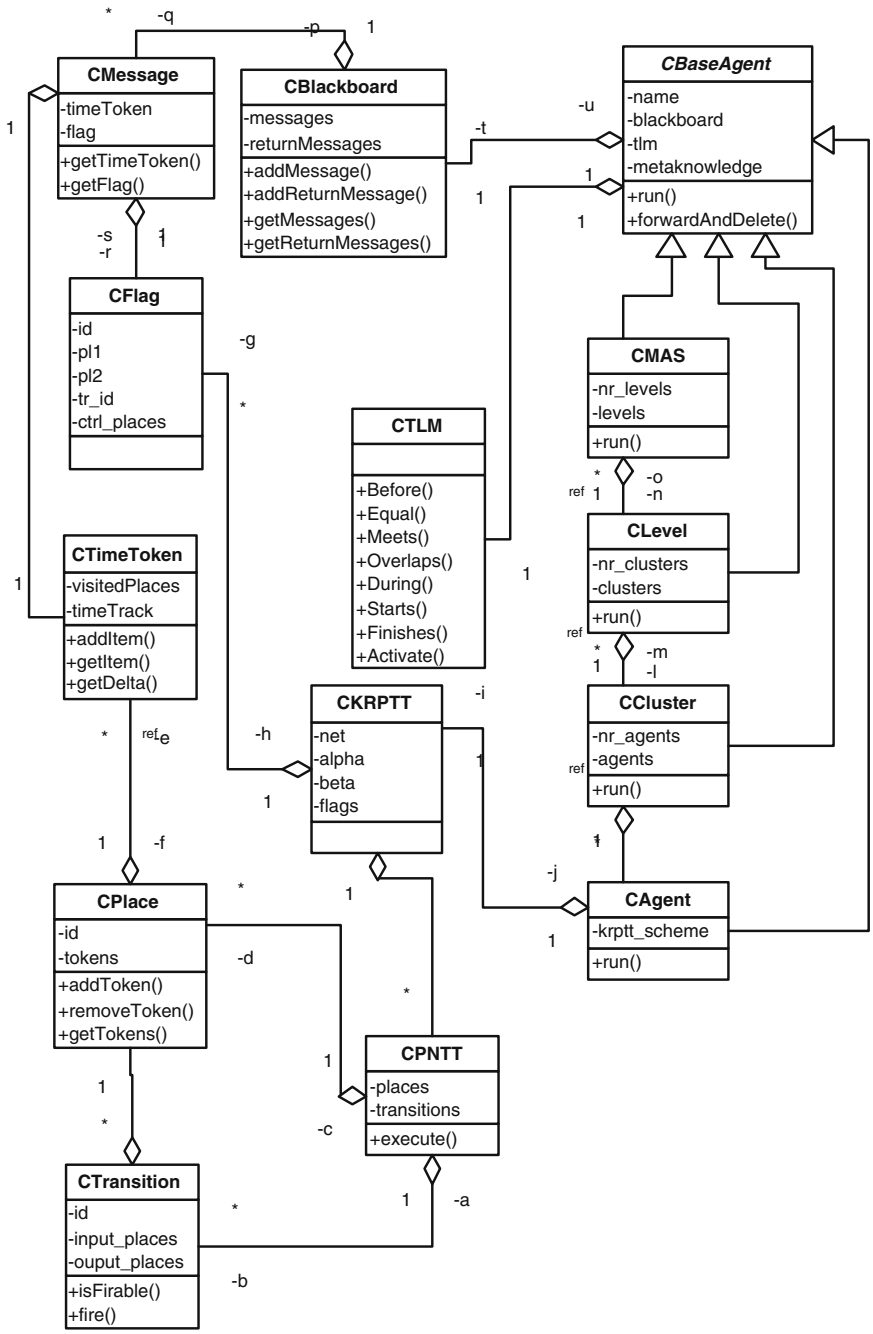

Fig. 11 UML class diagram of the object-oriented implementation of the simulator

Besides the instances of a blackboard, TLM, and meta-knowledge inherited from the class *CBaseAgent*, the main property of the class *CMAS* is a list of levels that the MAS contains. It is implemented as a list of instances of a class *CLevel*.

Similarly, the class *CLevel* contains a group of clusters that are represented as a list of instances of a class *CCluster*. The organization of the class *CCluster* is identical to the classes mentioned earlier and contains a list of instances of the class *CAgent*.

The class *CAgent* also inherits *CBaseAgent* in order to provide the agent's local blackboard, TLM, and meta-knowledge. But instead of containing a list of lower-level entities, an agent contains a knowledge base represented by means of a KRPTT knowledge representation scheme. In our object-oriented model, this scheme is represented by the class *CKRPTT*.

The class *CKRPTT* reflects the structure of the KRPTT knowledge representation scheme. It contains an instance of a Petri net with time tokens, a set of flags, and functions α and β, which give semantic interpretation to places and transitions. A PNTTs is also built from well-defined elements, i.e., places and transitions, and therefore its abstraction into a class is easy. In our implementation, it is represented by the class *CPNTT*. Places are represented as instances of the class *CPlace* and transitions as instances of the class *CTransition*. The class *CPlace* contains a list of tokens that are present in a specific place at a given moment of time.

Time tokens are represented by the class *CTimeToken*, the main property of which is a list of pairs (identifier (ID) of visited place, time of detainment of the token in that place). This class also contains an initial time of detainment of the token and the total accumulated time.

The basic property of the class *CTransition* is a list of pointers to all the input places and a list of pointers to all the output places. An important member function of *CTransition* is the function *fire()*, which fires a transition if it is enabled (i.e., removes time tokens from input places and puts them into output places of a transition, adding a new entry to a token's list of visited places).

Flags are realized as instances of the class *CFlag*, which contains indexes of two places for which a temporal relation has to be evaluated, an identifier of the mentioned temporal relation and a list of indexes of places into which a control token has to be put if the relation is to be satisfied.

5.2 TeMAS simulator description and user interface

Based on the hierarchical structure of the MAS described earlier, the KRPTT knowledge representation scheme and the underlying PNTT, the TeMAS simulator provides the means for describing the structure of the MAS, and the agent's knowledge base that describes situations from temporally rich domains, and it supports temporal reasoning. The program is developed in a C++ environment for Windows and Linux platforms. It has an open architecture and a user-friendly graphical interface. The main window of the program with drop-down menus and a toolbar is shown in Fig. 12.

The left-hand part of the main window shows the hierarchical structure of the currently simulated MAS in a tree-like form. The right-hand part of the window is a workspace where windows of different components of the sys-

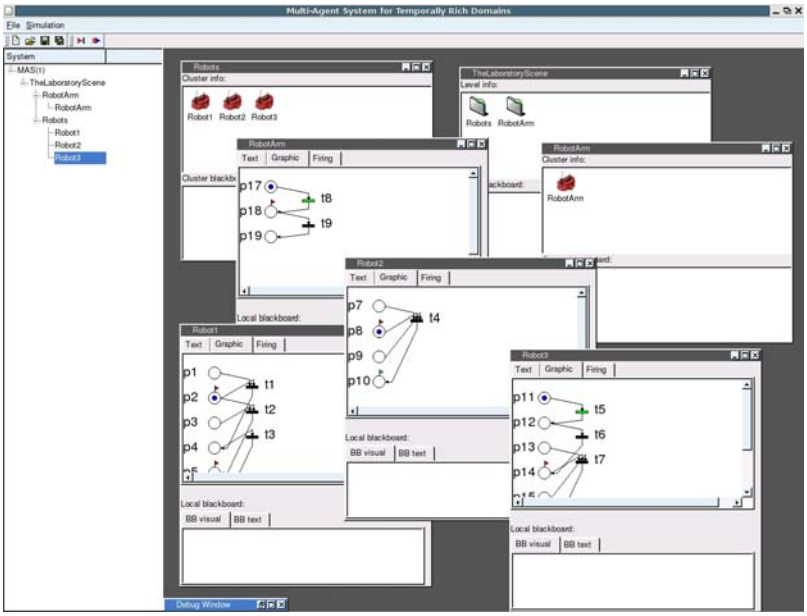

Fig. 12 The main window of the TeMAS simulator

tem represented by TeMAS (i.e., agents, clusters, levels, etc.) are displayed. The windows of the agents show the structure and marking of the agent's PNTTs, as well as the current state of the agent's local blackboard. The windows of other components (clusters, levels, and a MAS) contain an area with a symbolic representation of their subcomponents and a component's blackboard.

The user can define the hierarchical structure of the TeMAS either by loading a file with a definition of the system, or by manually adding different components via drop-down menus. When the structure of the TeMAS and all its subcomponents have been specified, the simulation can be performed either step-by-step or at once. If a goal state of the system is reached, or if the predefined number of steps is achieved, the simulation finishes and the simulator notifies the user about the result.

5.3 Example of using the TeMAS simulator for the interpretation of a dynamic scene

In this section, we give an example of using the TeMAS simulator to interpret a laboratory dynamic scene. The initial position of four agents (*Robot1*, *Robot2*, *Robot3*, and *Robot arm*) is shown in Fig. 13. *Robot1* and *Robot2* are equipped with sonars, and *Robot3* has a CCD camera. The *Robot arm* holds the brick. Three movable robots share a common goal: one of them has to reach the charger (Fig. 13).

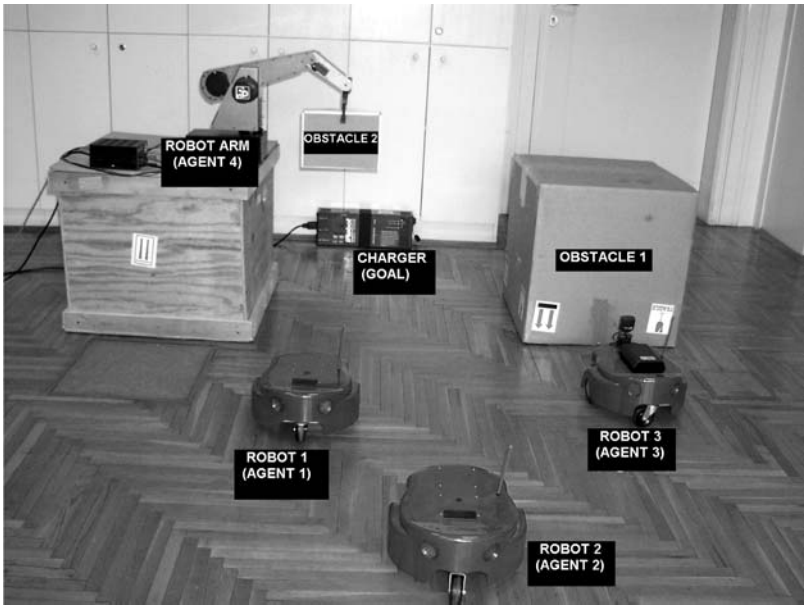

Fig. 13 Initial position of the agents

Fig. 14 Five frames taken from the dynamic laboratory scene

This goal has to be achieved in the shortest possible time (let us suppose that all three robots have the same velocity). At a glance, with the assumption that all the movable robots start to operate at the same time, *Robot3* is the nearest to the charger, but due to an obstacle (*Obstacle1*), its path is the longest. *Robot1* is the candidate for achieving the goal in the shortest possible time, but there is also the *Robot arm* that can drop a brick (*Obstacle2*) in the path of *Robot1*, making its mission impossible. Figure 14 shows five frames taken from the frame sequence of the dynamical scene.

At the event-recognition level of a hierarchical model of a computer-vision system (Fig. 5), the described situation is represented by the corresponding TeMAS level, consisting of four agents organized in two clusters (Fig. 15). Each agent is modeled by a corresponding PNTT. The time delays $\tau(p_i) = \tau_i$ as well as the function α are denoted at each place, while the function β is assigned to each transition. The initial marking Ω_0, the initial time accumulation of time tokens, and the distribution of the flags are also depicted in Fig. 15.

The time dependencies among the agent's activities are modeled by means of flags. The set of flags F for the modeled system is

$$F = \{f_1, f_2, f_3, f_4, f_5, f_6, f_{G1}, f_{G2}, f_{G3}\}.$$

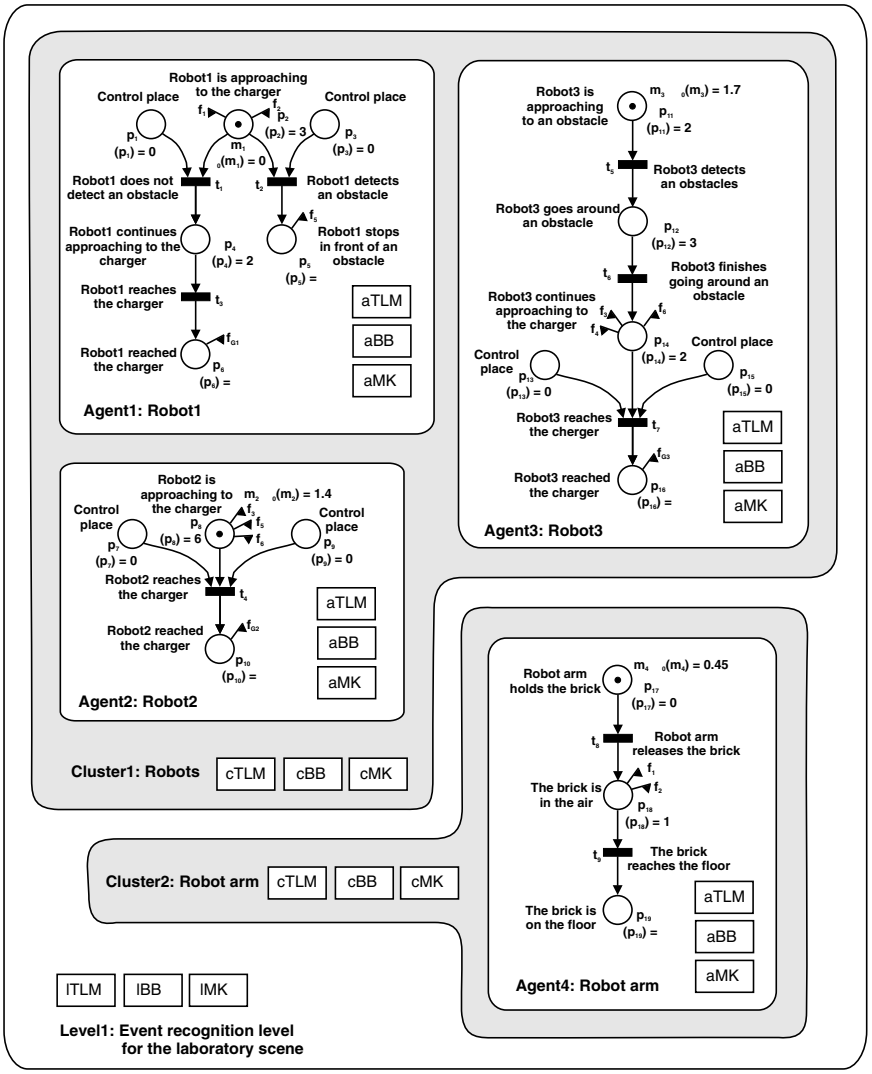

Fig. 15 TeMAS event-recognition level of the dynamic laboratory scene

The individual flags are defined as follows:

$f_1 = (p_2, p_{18}, tr_1, p_1)$, where $tr_1 =$ " $<$ OR d OR m OR o OR s";

$f_2 = (p_2, p_{18}, tr_2, p_3)$, where $tr_2 =$ " $>$ OR f OR $=$ OR di OR fi OR mi OR oi OR si";

$f_3 = (p_8, p_{14}, tr_3, p_{13})$, where $tr_3 =$ " $=$ OR f OR $>$ OR di OR fi OR mi OR oi OR si";

$f_4 = (p_5, p_{14}, tr_4, p_{15})$, where $tr_4 =$ " $<$ OR d OR $=$ OR f OR m OR o OR s OR $>$ OR di OR fi OR mi OR oi OR si";

$f_5 = (p_5, p_8, tr_5, p_7)$, where $tr_5 =$ " $<$ OR d OR $=$ OR f OR m OR o OR s OR $>$ OR di OR fi OR mi OR oi OR si";

$f_6 = (p_8, p_{14}, tr_6, p_9)$, where $tr_6 =$ " $<$ OR d OR m OR o OR s";

$f_{G1} = (p_6, -, -, -)$;

$f_{G2} = (p_{10}, -, -, -)$;

$f_{G3} = (p_{16}, -, -, -)$.

For example, the semantic of the flag f_1 can be interpreted as follows: if *Robot1* leaves the collision zone before the brick falls on the floor, then *Robot1* can continue approaching the charger. The flags $f_{Gi}; i = 1, 2, 3$ denote the goal states. For example, a flag f_{G2} denotes the goal state "*Robot2* has reached the charger".

The TeMAS simulator can generate the answers for different inquires. For example: Let us suppose that *Robot1* starts its moving at the time $t_0 = 0$ (the action starts at the beginning of the world), *Robot2* starts its moving after 1.4 time units and *Robot3* starts to operate after 1.7 time units. After 0.45 time units, *Robot arm* releases the brick. What will occur in the scene for the scenario mentioned earlier?

The intermediate and final results of the simulation of the scenario described earlier, performed step-by-step, are presented as follows. The screen snapshot of the TeMAS simulator with the graphical PNTT representation of the agents, the initial markings of the PNTTs, and the initial structures of the time tokens in separate windows (gray ones) is shown in Fig. 16.

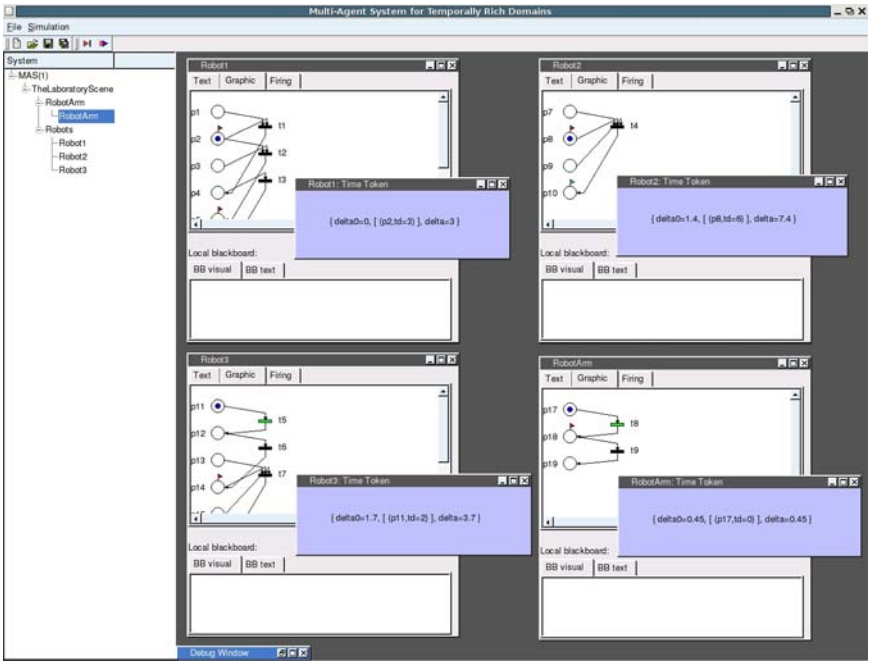

Fig. 16 Snapshot of the TeMAS simulator before the first step of the simulation

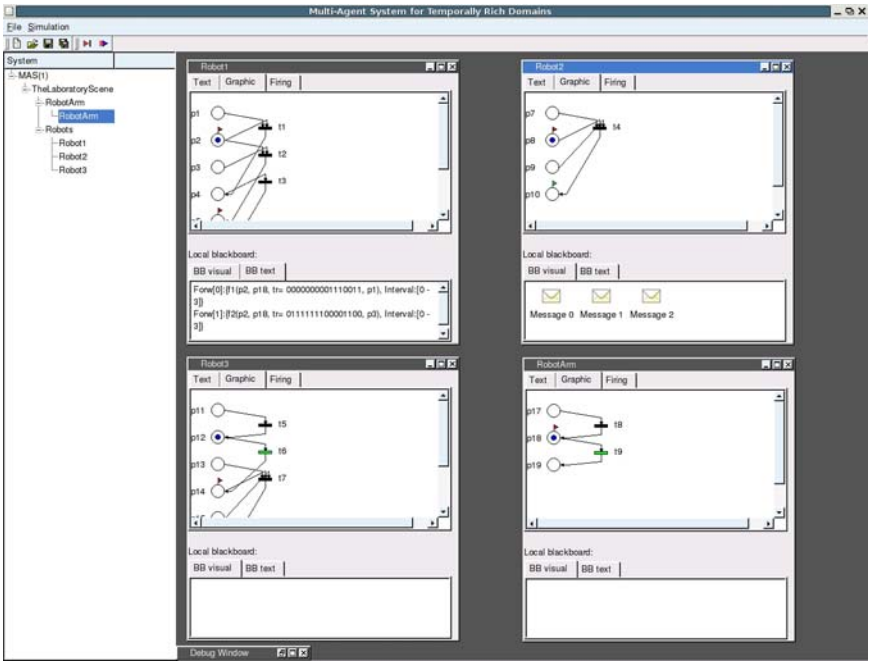

Fig. 17 The snapshot of the TeMAS simulator after the first step of the simulator

Since there is the time token m_1 in the place p_2 (belonging to agent *Robot1*) and the time token m_2 in the place p_8 (belonging to agent *Robot2*), which are marked by flags, messages will appear on the blackboards of these two agents after the first step of the simulation. Messages on the blackboard of *Robot1* are invoked by flags f_1 and f_2, while messages on the blackboard of *Robot2* are invoked by flags f_3, f_5, and f_6. The messages are represented by envelopes (graphical mode) or by a detailed textual description (Fig. 17). The first step will also result in the firing of enabled transitions t_5 (agent *Robot3*) and t_8 (agent *Robot arm*), resulting in new markings of the PNTTs of *Robot3* and *Robot arm*. The screen snapshot of the simulator after performing the first step of the simulation is shown in Fig. 17.

In the next step, all five messages from the blackboards of *Robot1* and *Robot2* are forwarded to a corresponding cluster blackboard (based on the meta-knowledge of corresponding agents), enabled transitions t_6 and t_9 are fired automatically, and two new messages are placed on the blackboard of the agent *Robot arm* due to the flags f_1 and f_2 placed at p_{18}. Since the testing places (p_2 and p_{18}) specified by flags f_1 and f_2 do not belong to the same cluster, messages corresponding to them are forwarded to the level blackboard. Messages invoked by flags f_3, f_5, and f_6 (located at the place p_8) will be evaluated by the cluster temporal logic module (cTLM) after the arrival of other corresponding messages, invoked by the flags f_3, f_5, and f_6 placed at p_{14} and p_5, to the cluster blackboard (cBB). In the subsequent steps, other messages are generated, forwarded and deleted, specified temporal relations for the time intervals are evaluated, return

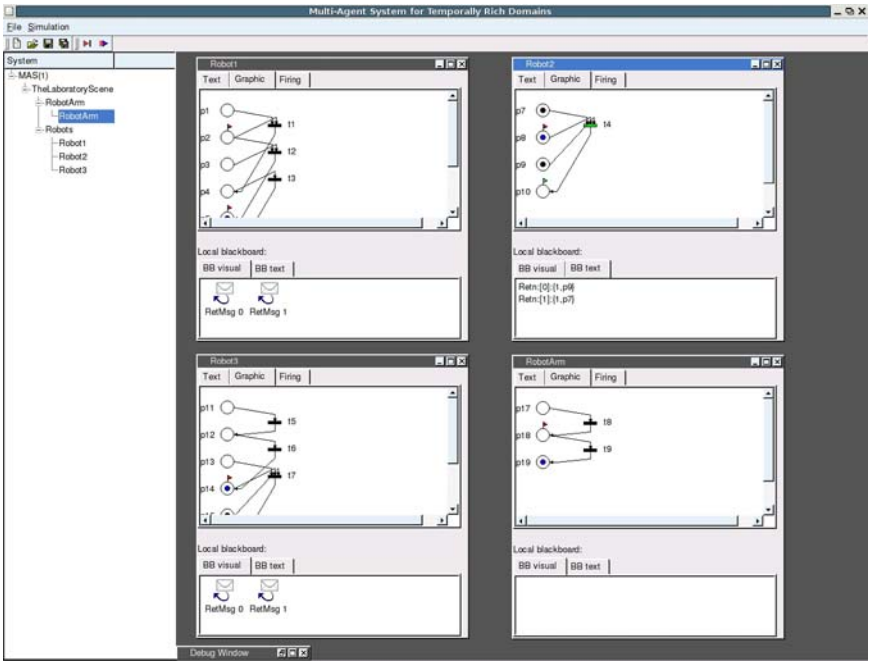

Fig. 18 The snapshot of the TeMAS simulator immediately before the last step of the simulation

messages are forwarded to the corresponding agent's blackboards, and depending on the result of the time relation evaluation (false or true), the control tokens are put in the control places. The situation immediately before the last step of the simulation is shown in Fig. 18. The blackboard of *Robot2* contains two return messages (depicted in Fig. 18 in textual mode), both of which specify the result of the evaluation "*true*." Therefore, control tokens are placed into the control places p_7 and p_9, and the transition t_4 is enabled.

In the last step of the simulation, the enabled transition t_4 fires and the time token reaches the place p_{10}. The place p_{10} is denoted by the goal flag f_{G2} and it corresponds to the goal state of the system: "*Robot2* has reached the charger." The snapshot corresponding to this final situation is shown in Fig. 19.

During the simulation, as well as after it is finished, temporal relations between different simulated activities can be analyzed by means of firing diagrams. The simulator generates a firing diagram for each agent automatically. The firing diagrams for four modeled agents are shown in Fig. 20. From the firing diagrams, it can be seen that *Robot1* enters the state "*Robot1* stops in front of an obstacle" (corresponding to the place p_5) in $t = 3.0$ time units, because the brick falls on the floor (place p_{19}) before *Robot1* reaches the collision zone, i.e., in $t = 1.45$ time units. The firing diagram related to *Robot2* also shows that *Robot2* reaches the charger in $t = 7.4$ time units. *Robot3* cannot reach the charger because *Robot2* is already there. In other circumstances (e.g., longer initial delay of *Robot2*), *Robot3* could reach the charger at $t = 8.7$ time units (Fig. 20).

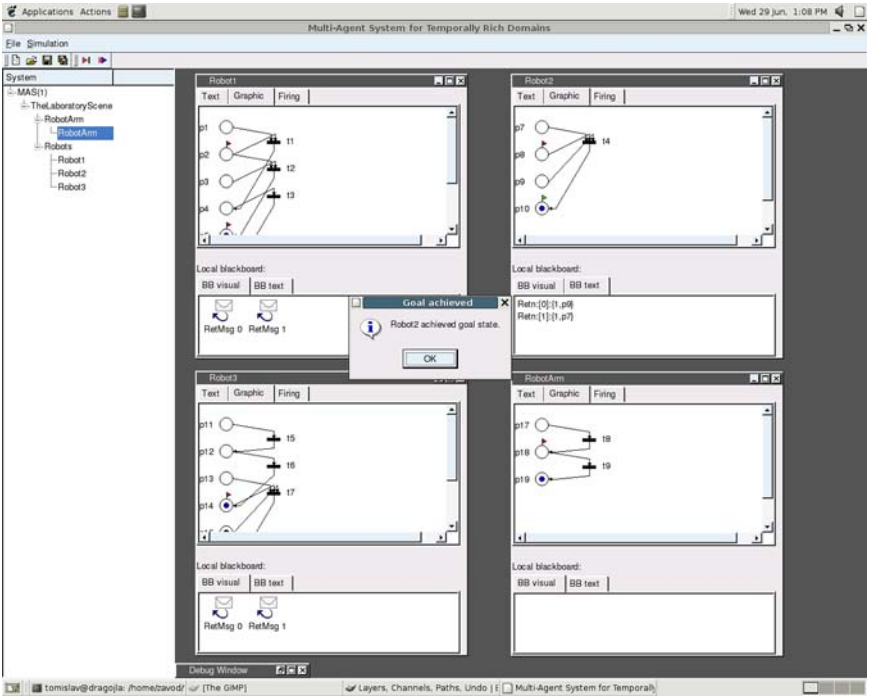

Fig. 19 The snapshot of the TeMAS simulator after the last step of the simulation

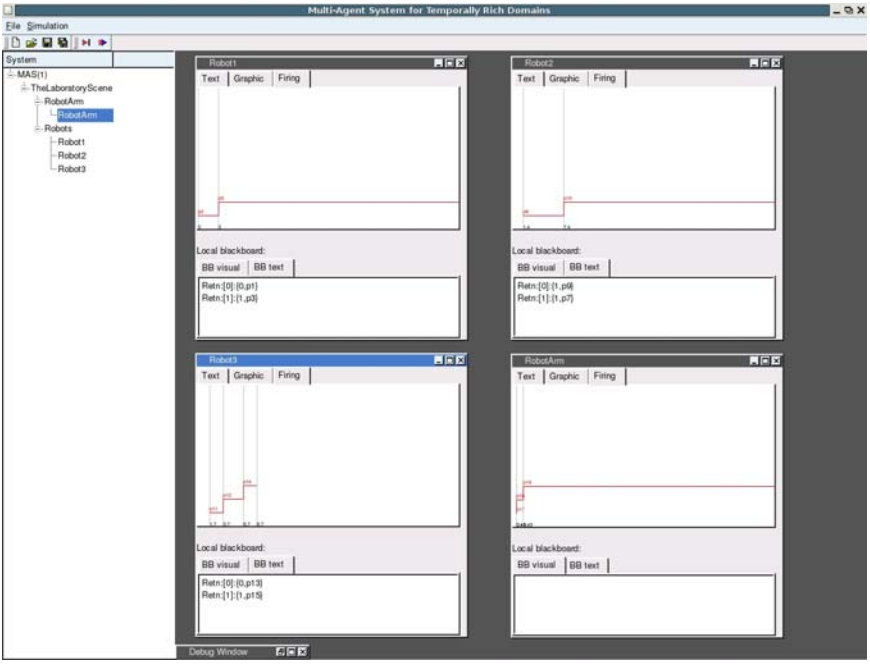

Fig. 20 Firing diagrams of simulated agents

By modifying the initial marking, the initial time of detainment of the time tokens, as well as by changing the time durations of some states and actions, different results for the simulation can be obtained.

6 Conclusion

The proposed MAS model called TeMAS was used for modeling and planning in temporally rich domains. The TeMAS effectively integrates temporal information and metric, logic between time primitives, action and/or state specification, and planning. The modified Petri nets, called PNTTs, are used as the basic building blocks for agent's knowledge representation and temporal reasoning, as well as for modeling situations in temporally rich domains.

The major advantages of the model are a unified representation of different temporal information, well-defined methods (based on the modified PN theory) for the analysis of the model, and simple modeling of the different time relations among the actions and states by changing the initial marking and time values assigned to time tokens and places.

Based on the hierarchical model of TeMAS, by using data-driven reasoning and the object-oriented simulator, the system can be used for analysis and planning actions and activities in temporally rich domains.

There are some obvious directions that will serve as a basis for our future work concerning the proposed model and simulator. First, we intend to extend the model in order to support an explicit representation of spatial information and then to develop an integrated spatio-temporal reasoning and planning model for hierarchical dynamic computer-vision systems.

Acknowledgements This work has been supported within the project "Multi-agent Systems for Dynamical Scene Interpretation," (no. 036025); Ministry of Science and Technology, Croatia. The authors would like to thank the anonymous reviewers for their invaluable suggestions and comments.

References

1. Albers M, Jonker CM, Karami M, Treur J (2004) Agent models and different user ontologies for an electronic market place. Knowl Inf Syst 6(1):1–41
2. Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843
3. Augusto JC, Simari GR (2001) Temporal defeasible reasoning. Knowl Inf Syst 3(3):287–318
4. Balan GC, Cioffi-Revilla C, Luke S, Panait L, Paus S (2003) MASON: a Java multi-agent simulation library. In: Proceedings of agent 2003 conference on challenges in social simulation, Chicago, IL
5. Barber F, Botti F, Onaindia E, Crespo A (1994) Temporal reasoning in REAKT (an environment for real-time knowledge-based systems). AiCommunications 7(3):175–202
6. Barrow HG, Tenenbaum JM (1981) Computational vision. Proc IEEE 69(5):72–579
7. Booch G, Rumbaugh J, Jacobson I (1999) The unified modeling language user guide. Addison-Wesley, Reading
8. Bruns G, Mossinger P, Polani D, Schmitt R, Spalt R, Uthman T, Weber S (2003) A simulation environment for continous virtual multi-agent systems, user manual. http://www.informatik.uni-mainz.de/ polani/XRaptor/XRaptor_7_3_8a/XRaptor_User_Manual-7.3.8a.pdf

9. Bulitko V, Wilkins DC (2003) Qualitative simulation of temporal concurrent processes using time interval Petri nets. Artif Intell 144:95–124
10. Cardoso J (1999) Time fuzzy Petri nets. Fuzziness in Petri nets. Springer-Verlag, Berlin, pp 115–145
11. Collier N (2001) RePast: an extensible framework for agent simulation. http://www.econ.iastate.edu/tesfatsi/RepastTutorial.Collier.pdf
12. Dean TL, McDermott DV (1987) Temporal data base management. Artif Intell 32:1–55
13. Decther R, Meiri I, Pearl J (1991) Temporal constraint network. Artif Intell 49:61–69
14. Dubois D, Prade H (1989) Processing fuzzy temporal knowledge. IEEE Trans Syst Man Cybernet 19(4):729–744
15. Ferber J (1999) Multi-agent systems: an introduction to distributed artificial intelligence. Addison-Wesley, Reading
16. He M, Leung H (2002) Agents in e-commerce: state of the art. Knowl Inf Syst 4(3):257–282
17. Helleboogh A, Holvoet T, Weyns D (2004) Time management adaptability in multi-agent systems. In: Proceedings of the fourth symposium on adaptive agents and multi-agent systems at the AISB '04 convention, Leeds, UKpp 20–29
18. Jensen K (1987) Coloured Petri nets. In: Lecture notes in computer science, vol 254. Springer-Verlag, Berlin, pp 248–299
19. Jensen K (1997) Coloured Petri nets. Springer-Verlag, Berlin
20. Jong W-T, Shiau Y-S, Horng Y-J, Chen H-H, Chen S-M (1999) Temporal knowledge representation and reasoning techniques using time Petri nets. IEEE Trans Syst Man Cybernet B 29(4):541–545
21. Jou SH, Kao SJ (2002) Agent-based infrastructure and an application to internet information gathering. Knowl Inf Syst 4(4):80–95
22. Kahn K, Gorry GA (1977) Mechanizing temporal knowledge. Artif Intell 9:87–108
23. Klein J (2002) BREVE: a 3D environment for the simulation of decentralized systems and artificial life. In: Proceedings of artificial life VIII, the 8th international conference on the simulation and synthesis of living systems, Wales, Sydney, Australia. The MIT Press, Cambridge, MA, pp 329–334
24. Klugl F, Puppe F (1998) The multi-agent simulation environment SeSAm. Paper presented at workshop simulation in knowledge-based systems, Paderborn, Germany. http://ki.informatik.uni-wuerzburg.de/ kluegl/pubs/1998/kluegl_puppe_SiWis98.pdf
25. Le Page C, Bousquet F (2000) CORMAS: a multiagent simulation toolkit to model natural and social dynamics at multiple scales. Paper presented at workshop the "ecology of scales", Wageningen. http://cormas.cirad.fr/pdf/cormasw.pdf
26. Lison M (2002) Deformation parameters in dynamic event recognition. In: Proceedings of the third international workshop on robot motion and control, RoMoCo '02, Bukowy, Poland. pp 325–330
27. Liu J, Tang YY (1999) Adaptive image segmentation with distributed behaviour-based agents. IEEE Trans PAMI 21(6):544–551
28. Logan B, Theodoropoulos G (2001) The distributed simulation of multiagent systems. Proc IEEE 89(2):174–185
29. Matarić MJ (2000) Getting humanoids to move and imitate. IEEE Intell Syst 15(4):18–24
30. Matarić MJ (2001) Learning in behavior-based multi-robot systems: policies, models and other agents. J Cogn Syst Res 2(1):81–93
31. McCharty J, Hayes PJ (1969) Some philosophical problems from the standpoint of artificial intelligence. In: Meltzer B, Michie D (eds) Machine intelligence, vol 4. Edinburgh University Press, Edinburgh, pp 463–502
32. Meimei G, Xiaoguang H, Zhiming W (2000) Linear logic as a tool for presentation and temporal reasoning of time Petri nets. In: Proceedings of the American control conference, Chicago, pp 3177–3181
33. Merlin P (1976) A methodology for the design and implementation of communication protocols. IEEE Trans Commun 24(6):614–621
34. Mesarovic MD, Macko D, Takahara Y (1970) Theory of multi-level hierarchical systems. Academic Press, New York
35. Minar N, Burkhart R, Langton C, Askenazi M (1996) The swarm simulation system: a toolkit for building multi-agent simulations. Santa Fe Institute Working Paper 96-06-042

36. Murata T, Suzuki T, Shatz SM (1999) Fuzzy-timing high-level petri nets (FTHNs) for time-critical systems. In: Cardoso J, Camargo H (eds) Fuzziness in Petri nets. Springer-Verlag, Heidelberg, pp 88–114
37. Palmer SE (1999) Vision science. The MIT Press, Cambridge, MA
38. Pelavin R, Allen JF (1986) A formal logic of plans in temporally rich domains. Proc IEEE 74(10):1364–1382
39. Peterson JL (1981) Petri net theory and modeling of systems. Prentice-Hall, Englewood Cliffs
40. Reisig W (1992) A primer in petri net design. Springer-Verlag, Berlin
41. Remagnino P, Tan T, Baker K (1998) Multi-agent vision surveillance of dynamic scenes. Image Vis Comput 16(8):529–532
42. Resnick M (1997) Turtles, termites and traffic jams: explorations in massively parallel microworlds. The MIT Press, Cambridge, MA
43. Ribarić S (2005) Temporal knowledge representation and reasoning model for temporally rich domains. In: Lecture notes in computer science: knowledge-based intelligent information and engineering systems, vol 3682. Springer-Verlag, Berlin Heidelberg New York, pp 430–436
44. Ribarić S, Dalbelo-Bašić B, Tomac D (2000) Object-oriented implementation of a model for fuzzy temporal reasoning. Proc Inf Process Manage Uncertain Knowl Based Syst 2:1247–1253
45. Ribarić S, Hrkać T (2003) Object-oriented simulator of multi-agent system for temporally rich domains. In: Proceedings of the 25th international conference on information technology interfaces, ITI 2003, Cavtat, Croatia, pp 397–402
46. Rus D (1998) Self-reconfiguring robots. IEEE Intell Syst 13(4):2–4
47. Russell SJ, Norvig P (1995) Artificial intelligence, a modern approach. Prentice-Hall, Upper Saddle River
48. Subrahmanian VS, Bonatti P, Dix J, Eiter T, Kraus S, Ozcan F, Ross R (2000) Heterogeneous agent systems. MIT Press, Cambridge, MA
49. Suzuki I, Lu H (1989) Temporal petri nets and their application to modeling and analysis of a handshake daisy chain arbiter. IEEE Trans Comput 38(5):696–704
50. Stavroulakis S, Callaghan V, Spacek L (2000) A multi-agent approach to machine vision. Paper presented at EXPO 2000: shaping the future, Hanover, Germany. http://www.shaping-the-future.de/pdf_www/051_paper.pdf
51. Symeonidis AL, Valtos E, Seroglou S, Mitkas PA (2005) Biotope: an integrated framework for simulating distributed multiagent computational systems. IEEE Trans Syst Man Cybern A Syst Hum 35(3):420–432
52. Vincent R, Horling B, Lesser V (2001) An agent infrastructure to build and evaluate multi-agent systems: the Java agent framework and multi-agent system simulator. In: Lecture notes in artificial intelligence: infrastructure for agents, multi-agent systems and scalable multi-agent systems, vol 1887. Springer-Verlag, Berlin Heidelberg New York, pp 102–127
53. Wang Y, Yuan B (2002) Fast method for face location and tracking by distributed behaviour based agents. IEEE Proc Vis Image Signal Process 149(3):173–178
54. Weiss G (1999) Multi-agent systems: a modern approach to distributed artificial intelligence. MIT Press, Cambridge
55. Weyns D, Holvoet T (2002) A Colored petri net for a multi-agent application. In: Proceedings of modeling objects, components and agents, Arhaus, Denmark, pp 121–140
56. Weyns D, Holvoet T (2004) A colored petri net for regional synchronization in situated multi-agent systems. In: Proceedings of the first international workshop on petri nets and coordination (PNC 04), Bologna, Italy, pp 65–85
57. Wooldridge M (2002) An introduction to multiagent systems. Wiley, Chichester
58. Yong T, Na T, XiaoPing Y, ZhiSheng F, Wei X (2003) An unified model of temporal knowledge and temporal data. In: Proceedings of the 8th international conference on computer-supported cooperative work in design, Xiamen, China, pp 711–713
59. Zaidi AK, Rizvi KH, Hussain SS (2003) On spatial modeling of discrete event systems using point-interval logic. In: Proceedings of the second international conference on machine learning and cybernetics, Xian, pp 1699–1704

Author Biographies

Slobodan Ribarić received the B.S. degree in electronics, the M.S. degree in automatics, and the Ph.D. degree in electrical engineering from the Faculty of Electrical Engineering, Ljubljana, Slovenia, in 1974, 1976, and 1982, respectively. He is currently a Full Professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. His research interests include pattern recognition, artificial intelligence, biometrics, computer architecture and robot vision. He has published more than 150 papers on these topics and authored four books (*Microprocessor Architecture*, *The Fifth Computer Generation Architecture*, *Advanced Microprocessor Architectures*, *CISC and RISC Computer Architecture*) and co-authored one book (*An Introduction to Pattern Recognition*). Dr. Ribarić is a Member of the IEEE, ISAI and IAPR.

Tomislav Hrkać received the B.S. degree in computer science from the Faculty of Electrical Engineering and Computing at the University of Zagreb, Croatia, in 1999. Since October 2000, he has been a Researcher with the Department of Electronics, Microelectronics, Computer and Intelligent Systems at the same faculty. He received the M.S. degree in 2004. As a co-author, he published several papers in international conference proceedings and a paper in a reviewed scientific journal. He is a Student Member of IEEE.