Object-Oriented Simulator of Multi-Agent System for Temporally Rich Domains

Slobodan Ribarić and Tomislav Hrkać

Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

An object-oriented implementation of a program simulator for a multi-agent system (MAS) is described. The model of a MAS is hierarchical model, consisting of different levels, where each level contains clusters of agents. A paradigm of a blackboard is used for communication among agents, clusters, as well as levels. The Petri Nets with Time Tokens are used as a basic concept for this model. An example of the use of the object-oriented simulator in a dynamic scene analysis is given.

Keywords: object-oriented implementation, multi-agent system, Petri nets, temporally rich domains.

1. Introduction

In recent years, multi-agent systems (MAS) have become an important issue in many fields of artificial intelligence, such as image interpretation and machine vision [13], [18], [22], distributed systems [10], robotics [14], and artificial life [15], [21].

The paper presents an object-oriented program implementation of a simulator of a hierarchical MAS for temporally rich domains. The aim of such simulator is its application in mobile object behavior analysis in dynamical computer vision scenes. These moving objects can be viewed as agents that simultaneously perform different (cooperative or antagonistic) tasks in time, and this is the reason why the scene is considered as temporally rich domain. According to Pelavin and Allen, temporally rich domains can be defined as domains which include concurrent actions that take time, the simultaneous occurrence of many actions at once, and domains with external events [16].

There are some recent works that address the problems of MAS simulation theory and simulators. Davila and Tucci introduced a logic-based multi-agent simulation theory [4]. Horling, Lesser and Vincent described a MAS simulation framework, the aim of which was realistic modeling of adaptive behavior in MASs and evaluation of different multi-agent coordination strategies [9]. Ferber developed a language called BRICK for description of MASs by Petri nets [7]. Klupsch proposed an object-oriented representation of time-varying data sequences in MASs [12]. Davila and Uzcagegui presented an object-oriented multi-agent simulation platform called GALATEA [5]. Esmin et al. also used object-oriented approach for their multi-agent simulation and educational tool for power system operation [6]. Object-oriented approach was also followed by Henoch and Ulrich, who presented an agent-based simulation platform for evaluating management concepts [8].

2. A Model of Hierarchical Multi-Agent System

2.1. Definition of a MAS

In general, a MAS can be defined as n-tuple [7]:

$$\text{MAS}=(\text{E, O, A, R, Op, LoU}),$$

where E is an environment, i.e. space which has a volume. In our experimental environment, E is a dynamical scene – a space with

defined metrics. By knowing physical characteristics of objects (velocity, acceleration), and because of metric space, a temporal component can be assigned to each action. O is a set of objects situated in E. The objects are movable and/or stationary and they can be perceived by agents. A is an assembly of agents. Agents may be represented as specific objects ($A \subseteq O$) representing active entities with or without sensors. R is a set of temporal and some spatial relations among objects and agents. Op is a set of operations of agents, such as: perceiving, transforming and manipulation of objects. LoU is a set of, so-called, laws of the universe, which are common for the environment E.

2.2. Hierarchical Organization of a MAS

It is well known that computer vision systems are naturally structured and represented as hierarchical systems [2]. According to abovementioned principle, the proposed MAS model has hierarchical organization. Each level of the MAS model consists of one or more agent clusters. Agents with identical or similar tasks are grouped into clusters. One of the reasons for such approach is emphasized communication among the agents in the cluster.

The main communication mechanism in the model is based on blackboard paradigm [3]. Each component of hierarchical organization (i.e. an agent, a cluster and a level) has an instance of a blackboard. At the top of hierarchical organization, there are global agent and the global blackboard, which support a communication among levels. When a communication between agents is needed (and that occurs in case when a certain relation between two time intervals has to be evaluated), it is performed as follows: Initially, an agent writes a message to its local blackboard. The message corresponds to one of the above mentioned time intervals. If the agent, according to its meta-knowledge, concludes that it will have sufficient information to inference about certain temporal relation, the message remains on its local blackboard and the agent waits for another message. Otherwise, the agent performs a "forward and delete" action: it forwards the message to the higher level (i.e. cluster) and deletes it from its local blackboard. The same procedure is repeated until the destination level is achieved.

2.3. Knowledge Representation and Reasoning

An agent has partial knowledge about the scene, meta-knowledge and ability of reasoning. These properties of the agent are represented by knowledge representation scheme based on Petri Nets with Time Tokens (PNTT) [19], [20].

The PNTT is a 8-tuple $(P,T,I,O,\tau,M,\upsilon,\Omega)$, where P,T,I,O are components of a generalized Petri net [18], τ is mapping from set of places to set of time delays, M is a set of time tokens, υ is a mapping called time accumulation assigned to each token and Ω is a marking of PNTT. A time token, similarly to the token in Colored PN [12], has its individuality – it carries information about the list of times of its detainment at all visited places. In PNTT, a transition is enabled if each of its input places has at least as many time tokens in it as arcs from the place to the transition *and* if the time of detainment of these tokens in places has elapsed. Such tokens are called "movable" time tokens. By firing a transition in the PNTT, the tokens are distributed to its output places. However, the firing changes the information of the corresponding time token, according to the time accumulation function υ.

Based on PNTT, a knowledge representation scheme KRPTT is defined as n-tuple [19]: $KRPTT=(PNTT,TLM,\alpha,\beta,F)$, where PNTT is a Petri net with time tokens, TLM is temporal logic module, α and β are bijective functions which assign a concept of states/actions or events to places and transitions of the PNTT, respectively. F is a set of flags. In general, flag $f_i \in F$ has the following structure: $(p_i, p_j, tr, p_l,...,p_m)$, where p_i and p_j denote places with time tokens that have to be tested by the TLM, according to the temporal relation tr, and $p_l,...,p_m$ are output places, in which the TLM sets control tokens depending on the results of evaluation of tr. These tokens can be treated as time tokens with zero accumulation time. There are also some types of degenerated flags [19].

Temporal reasoning is based on time tokens, and temporal logic module (TLM) which implements Allen's temporal logic [1].

3. Object-Oriented Implementation of a Simulator

Object-oriented design includes the following elements: abstraction into classes and objects, encapsulation, modularity and inheritance with polymorphism.

A MAS structure is built from well-defined elements, with proper relationships; therefore, abstraction of these elements into classes is straightforward. Four main classes are used to build this structure (Fig. 1): *CMAS*, *CLevel*, *CCluster* and *CAgent*, which represent different components of our MAS model: a MAS, a level, a cluster and an agent, respectively. The mentioned hierarchical levels share a similar structure: each of them contains a blackboard, a temporal logic module and either a set of lower level entities (for example, MAS contains levels, level contains clusters and cluster contains agents) or a knowledge base in form of a KRPTT (if the component is an agent).

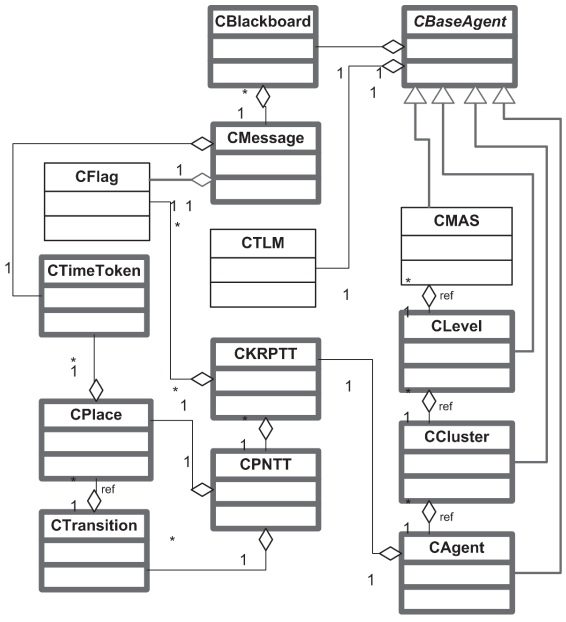

Fig. 1. Object-oriented implementation of the simulator.

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

CBaseAgent class contains an instance of a class *CBlackboard* and an instance of a class *CTLM*. 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, and for locking and unlocking a blackboard in order to provide exclusive access to it. Messages are implemented as instances of a class *CMessage*.

CMessage reflects a message structure and is composed of a time token and a corresponding flag. *CTLM* class represents a temporal logic module, which is responsible for evaluation of temporal relations.

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

Similarly, *CLevel* contains a group of clusters which are represented as a list of instances of a class *CCluster*. The organisation of *CCluster* is identical to the above-mentioned classes and it contains a list of instances of *CAgent* class.

CAgent also inherits *CBaseAgent* in order to provide the agent's local blackboard and TLM. But, instead of containing a list of lower-level entities, an agent contains knowledge represented in form of a KRPTT knowledge representation scheme [19]. This scheme is in our object-oriented model represented by *CKRPTT* class.

CKRPTT reflects the structure of KRPTT knowledge representation scheme. It contains an instance of Petri net with time tokens, a set of flags, and functions α and β which give semantic meaning to places and transitions.

A Petri net with time tokens is also built from well-defined elements: places and transitions and therefore its abstraction into class is easy. In our implementation it is represented by class *CPNTT*. Places are represented as instances of *CPlace* and transitions as instances of *CTransition* class. Class *CPlace* contains a list of tokens which are present in specific place at given moment of time.

Time tokens are represented by *CTimeToken* class, which main property 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 total accumulated time.

The basic property of class *CTransition* is a list of pointers to all input places and a list of pointers to all output places. An important member function of *CTransition* is 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 into a token's list of visited places).

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

4. Program Description

Based on the described hierarchical structure of the MAS, KRPTT knowledge representation scheme and underlying PNTT, the program provides the means for describing a structure of the MAS, agent's knowledge base that describes situations from temporally rich domains and it supports temporal reasoning. It is developed in a C++ environment for Windows and Linux platforms. It has an open architecture and an user-friendly graphical interface. The main window of this program is shown in Figure 2. In its upper part, there are drop-down menus and toolbar.

The main window area is divided in two parts. The left part shows a hierarchical structure of the currently simulated MAS in a tree-like view. The right part of the main window is a workspace where windows of different components of the system (i.e. agents, clusters, levels etc.) are displayed.

The user can define hierarchical structure of the MAS either by loading a file with definition of the system, or by manual adding different components via drop-down menus.

5. An Example

In this section we give an example of a laboratory dynamic scene and its simulation. The initial position of four agents (*robot1*, *robot2*, *robot3* and *robot arm*) is shown in Fig. 3.

Robot1 and *robot2* are equipped with sonars, and *robot3* has a CCD camera. Three movable

Fig. 2. The main window of the simulator.

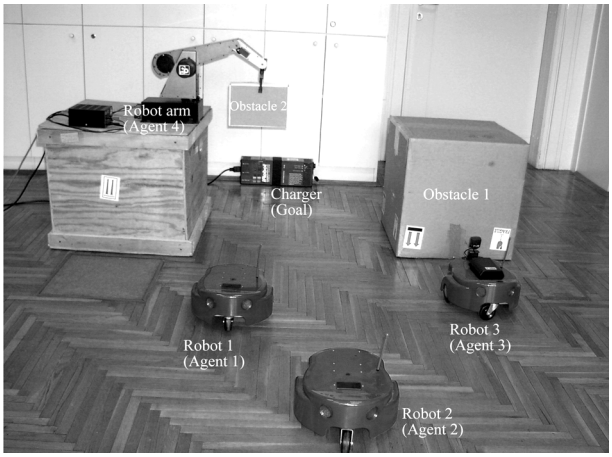

Fig. 3. Initial position of agents.

Fig. 4. Five frames taken from dynamic scene.

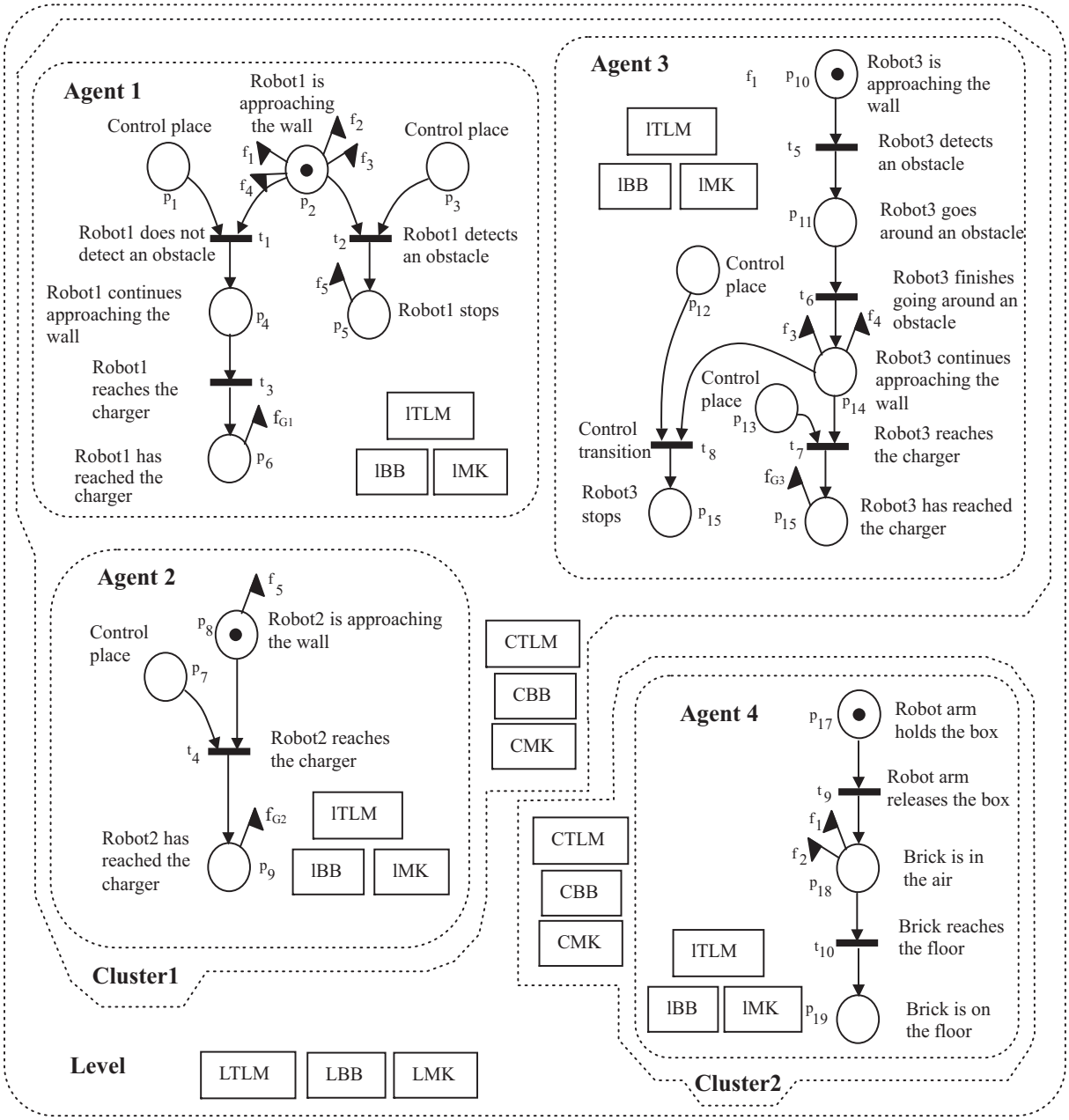

Fig. 5. A model for described example.

robots share a common goal: one of them has to reach the charger (Fig. 3).

This goal has to be achieved in the shortest possible time (let us suppose that all three robots have the same velocity). Figure 4 shows five frames taken from the frame sequence of a dynamical scene. At a glance, *robot3* is the nearest to the charger, but due to an obstacle (*obstacle1*), its path is the longest. The *robot1* is the candidate for achieving the goal in the shortest possible time. There is also a *robot arm* that can drop a box (*obstacle2*) on the way of *robot1*, making its path longer than that of *robot2*. This situation can be represented as one level of the MAS model and it is shown in Figure 5.

The result of the simulation of the situation described above is shown in Fig. 6. The simulation shows that *robot2* will reach the charger.

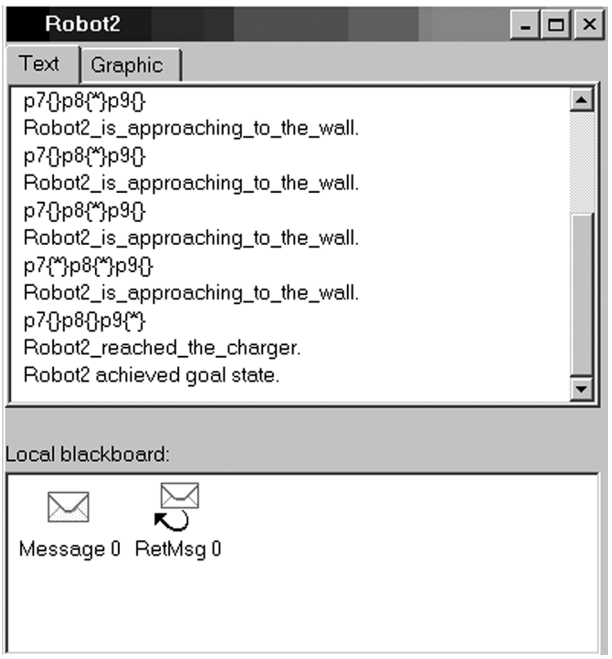

Fig. 6. The result of the simulation.

6. Conclusion

Object-oriented implementation of a simulator of MAS for temporally rich domains is described in the paper. The simulator is designed for hierarchical MAS, which contains multilevel structure, where each level consists of clusters and each cluster is built of agents. The modified Petri nets called Petri Nets with Time Tokens are used as basic building blocks for agent's knowledge representation and temporal reasoning, as well as for modelling situations in temporally rich domains.

References

[1] ALLEN J.F., Maintaining Knowledge about Temporal Intervals, *Communications of the ACM* 1983, Vol. 26, No. 11, pp. 832–843.

[2] BARROW H.G., TENENBAUM J.M., Computational Vision, in *Proceedings of the IEEE*, Vol. 69, No. 5, May 1981, pp. 572–579.

[3] CARVER N., LESSER V., *The Evolution of Blackboard Control Architecture*, CMPSCI Technical Report 92–71, 1992.

[4] DAVILA J., TUCCI K., *Towards a Logic-Based Multi-Agent Simulation Theory*, International Conference on Modelling, Simulation and Neural Networks MSNN 2000.

[5] DAVILA J., UZCAGEGUI M., *GALATEA: A Multi-Agent Simulation Platform*, International Conference on Modelling, Simulation and Neural Networks MSNN 2000.

[6] ESMIN A., AOKI A., LOPES C., LAMBERT-TORRES G., Multi-Agent Simulation and Educational Tool for Power System Operation, *Proceedings of the VII International Conference on Engineering and Technology Education INTERCOM* 2002.

[7] FERBER J., *Multi-Agent Systems: An introduction to Distributed Artificial Intelligence*, Adison-Wesley, 1999.

[8] HENOCH J., ULRICH H., Agent-Based Simulation Platform for Evaluating Management Concepts, *Proceedings of EUROSIM* 2001.

[9] HORLING B., LESSER V., VINCENT R., Multi-Agent System Simulation Framework, in *16th IMACS World Congres 2000 on Scientific Computation, Applied Mathematics and Simulation*, August 2000.

[10] JAMALI N., THATI P., AGHA G.A., An Actor-Based Architecture for Customizing and Controlling Agent Ensemblies, *IEEE Intelligent Systems*, 1999, pp. 38–44.

[11] JENSEN K., Coloured Petri Nets, Lecture Notes in *Computes Science*, No. 254, Springer-Verlag, Berlin, 1987, pp. 248–299.

[12] KLUPSCH M., Object-Oriented Representation of Time-Varying Data Sequences in Multi-Agent Systems, in Nagib C. Callaos, editor, *World Multiconf. on Systemics, Cybernetics and Informatics – 4th International Conference on Information Systems, Analysis and Synthesis*, Vol. 2, 1998, pp. 33–40.

[13] LIU J., TANG Y.Y., Adaptive Image Segmentation With Distributed Behaviour-Based Agents, *IEEE Trans. on PAMI*, Vol. 21, No 6, June 1999.

[14] MATARIĆ M.J., Getting Humanoids to Move and Imitate, *IEEE Intelligent Systems*, 2000, pp. 18–24.

[15] MATARIĆ M.J., Learning in Behavior-Based Multi-Robot Systems: Policies, Models and Other Agents, *Journal of Cognitive Systems Research* 2, 2001, pp. 81–93.

[16] PELAVIN R., ALLEN J.F., A Formal Logic of Plans in Temporally Rich Domains, in *Proceedings of the IEEE*, Vol. 74, No. 19, 1985, pp. 1365–1382.

[17] PETERSON J.L., *Petri Net Theory and Modeling of Systems*, Prentice-Hall, Englewood Cliffs, 1981.

[18] REMAGNINO P., TAN T., BAKER K., Multi-Agent Visual Surveillance of Dynamic Scenes, *Image and Vision Computing*, 16 (1998), pp. 529–532.

[19] RIBARIĆ S., DALBELO BAŠIĆ B., Temporal Knowledge Representation and Reasoning Model Based on Petri Nets with Time Tokens, in *Procedings of Industrial Applications in Power Systems, Computer Science and Telecommunications*, Vol. 1, 1996, pp. 131–135.

[20] RIBARIĆ S., HRKAĆ T., PAVEŠIĆ N., Hierarchical Model of Multi-Agent System for Spatio-Temporal Rich Domains, *Proceedings of the 6th International Conference on Intelligent Engineering Systems INES* 2002, pp. 321–326.

[21] RUS D., Self-Reconfiguring Robots, *IEEE Intelligent Systems*, July/August 1998, pp. 2–4.

[22] STAVROULAKIS S., CALLAGHAN V., SPACEK L., *A Multi-Agent Approach to Machine Vision*, EXPO 2000: Shaping the Future, 2000.

Received: June, 2003
Accepted: September, 2003

Contact address:
Slobodan Ribarić, Tomislav Hrkać
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3
10000 Zagreb
Croatia
e-mail: slobodan.ribaric@fer.hr
tomislav.hrkac@fer.hr

SLOBODAN RIBARIĆ received his B.S. degree in electronics, the M.S. degree in automatics, and the PhD. 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, computer architecture and robot vision. He has published more than one hundred and twenty papers on these topics, and he is the author of four books (*Microprocessor Architecture*, *The Fifth Computer Generation Architecture*, *Advanced Microprocessor Architectures*, *CISC and RISC Computer Architecture*) and the co-author of one (*An Introduction to Pattern Recognition*). Dr. Ribarić is a member of the IEEE and IAPR.

TOMISLAV HRKAĆ received his B.Sc. in computing from the University of Zagreb in 1999. He has been a researcher at the Department of Electronics, Microelectronics, Computer and Intelligent Systems since September 2000. He is the co-author of several papers published in international conference proceedings.