Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 16, No. 2, June 2016, pp. 95-103
http://dx.doi.org/10.5391/IJFIS.2016.16.2.95

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

An Adaptive Goal-Based Model for Autonomous Multi-Robot Using HARMS and NuSMV

Yongho Kim[1], Jin-Woo Jung[2], John C. Gallagher[3], and Eric T. Matson[1]
[1]M2M Lab, Computer and Information Technology, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science and Engineering, Dongguk University, Seoul, Korea
[3]Department of Computer Science and Engineering, Wright State University, Dayton, OH, USA

ijfis

Abstract

In a dynamic environment autonomous robots often encounter unexpected situations that the robots have to deal with in order to continue proceeding their mission. We propose an adaptive goal-based model that allows cyber-physical systems (CPS) to update their environmental model and helps them analyze for attainment of their goals from current state using the updated environmental model and its capabilities. Information exchange approach utilizes Human-Agent-Robot-Machine-Sensor (HARMS) model to exchange messages between CPS. Model validation method uses NuSMV, which is one of Model Checking tools, to check whether the system can continue its mission toward the goal in the given environment. We explain a practical set up of the model in a situation in which homogeneous robots that has the same capability work in the same environment.

Keywords: Adaptive model, Multi-robot systems, Model checking, HARMS, NuSMV

1. Introduction

Use of automated robots has been shifted from industry to human-friendly environments such as home, office, school, and public places. Because automated robots are now exposed to a new dynamic environment rather than a static workspace, such robots face many uncertainties and have to consider dynamic environmental factors as well as their tasks in order for them to be autonomous. This means that the robots should be able to have additional functions (e.g., make an organization to work together [1], adjust the system model to continue working [2]) to deal with such factors to accomplish their mission. According to the study [3], one of the challenges in ubiquitous robotics in complex environment is to discern a situation and perform a reasonable reaction upon the situation.

When autonomous robots encounter an uncertainty while performing a task, the problem becomes real. Such uncertainties might not be crucial, which means that the uncertainties do not interrupt a robot in terms of the attainment of the goal. For example, an autonomous ground vehicle is driving on a road. The vehicle could encounter some obstacles on the road. If the vehicle has an obstacle avoidance function, it could overcome the uncertainty and drives to its destination to complete the given task. The vehicle could even keep driving over the obstacle if it does not have the function. However, if one of the tires is punctured by a sharp

Received: Jun. 2, 2016
Revised : Jun. 20, 2016
Accepted: Jun. 21, 2016

Correspondence to: Yongho Kim
(kim1681@purdue.edu)

object and becomes at while driving, the vehicle could not be able to arrive to the destination and the task could not be accomplished. Because this change is critical in terms of reliability of the robot, designers should be able to address those changes even after they deploy the robot.

The concept of adaptive model is that a model accepts changes and evolves toward the changes. For autonomous robots, this charming ability allows them to continue their working whenever they encounter unexpected changes. The adaptive model in the study [4] helps agents reorganize their groups based on agents capabilities at run-time. Because the authors focused on organizational aspect, they do not change their original model. Instead, they rearrange agents to meet the requirement. Adaptive concept is applied to not only physical changes, but also software side changes in the field of robotics. R-object model [5] allows robots to adapt to the environment by relinking and reconstructing task schedules with regard to the current status of robots.

In multi-robot environment, robots share the workspace. A robot recognizes other robots as environmental entities. From a robot's view seeing such environment, complexity of analyzing the environment increases as number of robot increases. When robots are doing the same thing (e.g., homogeneous robots), chance of conflict becomes larger. This phenomenon also indicates that the robot has to have an ability to adapt its model to the given multi-robot environmental condition. One of the simpler ways to avoid conflicts is to apply a simple rule-based algorithm to all agents [6]. On the other hand, another way to have such ability would be to establish a communication network among robots, which enables them to transmit environmental information to others. Matson and Min [7] introduced Human-Agent-Robot-Machine-Sensor (HARMS) model for interactions among heterogeneous actors. HARMS connects actors over network by peer-to-peer manner and uses particular message types such that all actors are indistinguishable in terms of which type of actor (e.g., robot, software agent, or even human) sends a message [8].

In order to bring the such adaptation ability to autonomous robots, we proposed an adaptive task-based model that keeps checking its model with the given stimuli coming from the environment [9]. When the stimuli change the model or significantly affect to the model, proposed model investigates the changes to check whether or not the system can continue the given task. The model also utilizes the HARMS model to tell other robots the changes that they observed. This enables the robots to rapidly adapt to the new environment. The previous research, however, lacks information about principles of multi-agent systems, which we believe is significantly important to design the model. Therefore, this paper an extension of the published research article focuses on justifying and clarifying the research problem of the previous article.

2. Related Work

2.1 Modeling Multi-Agent Systems

Most recent active research studies of modeling multi-agent systems can be boiled down to agent-oriented paradigm (AOP) [10–14] and agent-oriented software engineering (AoSE) [15–17]. In both AOP and AoSE, software agents are modeled not only for their properties and functionality, but also for interfaces and message handling capabilities. This is a very different approach from a classic object-oriented paradigm (OOP) because AOP and AoSE focus more on relationships and interplays between agents. Moreover, an agent model has a kind of reasoning capability to determine what to do, whereas an object just does it for free [17].

The concept of holonic system has been proposed by Rodriguez et al. [18] and implemented in a programming language, called 'SARL' [14]. The SARL is powered by Eclipse IDE and is rooted in AOP. The SARL allows to design complex systems in which the robots may frequently deal with interactions, concurrency, and distributions.

Similarly, the organization model for adaptive computational systems (OMACS) model has been proposed by Deloach et al. [4]. As the name implies the model considers potential changes of the structure of the system in run-time, based on how the agents perform their capabilities to play roles that satisfy the goals. The agents can be reassigned to new roles at run-time when the result of the system analysis finds more efficient way to fulfill the goals. This makes the system structure flexible.

JaCaMo [13] is a programming platform of the framework highlighting the three dimensions: *organization*, *agent*, and *environment*. JaCaMo well reflects AOP and further extends multi-agent oriented paradigm (MAOP). In contrast with the *holonic* structure, JaCaMo framework supports indirect communication among the agents through artifacts in the environment model.

The proposed model is very different from the aforementioned models in terms of strategy of how to react upon changes. The aforementioned models are trying to adapt themselves to the environment by changing structures and links of the current organization, while the proposed model changes the system

model by itself. Because a robot now can change what it can do and what it cannot do, its capability also changes. However, it is similar with the R-object model in that the R-object defines what a robot should do and what the robot can do, named 'platform independent function (PIF)' and 'platform dependent function (PDF),' respectively.

2.2 Model Verification in Robotics

The model checking technique [19] has been used to check systems that require a system level verification due to its complexity [20, 21]. The model checking technique requires two specifications: a system model and the system requirements. The system model can be described using a finite state machine (FSM). A FSM consists of states that represent status of the system and transitions which describe what and how a transition is made. System requirements are expressed in a linear temporal logic (LTL). Because the system model is only composed of logical states and transitions, model checking techniques check only logical abnormality based on the system requirements (i.e., LTLs).

Although model checker requires massive computational resources, many research studies utilized it for their systems. In particular, the technique has been highlighted in the field of robotics from about a decade ago because autonomous robots become more complex and the capability of the robots for attainment of goals should be verified in order to improve reliability of the robots. Fainekos et al. [22] defined motions of a mobile robot using LTL. Konur et al. [23] proposed a probabilistic model checking for a swarm robots. Goppert et al. [24] modeled a flapping-wings micro air vehicle (FWMAV) using polyhedral invariant hybrid automation (PIHA) to check if the vehicle could accomplish the mission with unpredictable turbulence. Kim et al. [25] proposed a training procedure of a humanoid soccer robot and verified the procedure using a model checker, which named 'NuSMV' [26].

3. Adaptive Goal-Based Model

Omicini et al. [11] described what are the components to model multi-agent systems. In fact, because most multi-agent systems models are from AOP, they follow the three fundamentals: agents, artifacts, and environments (or workspace). Here are the short definitions of the fundamentals:

- *Agents*: to represent pro-active components of the systems, encapsulating the autonomous execution of some kind of activities inside some sort of environment
- *Artifacts*: to represent passive components of the systems such as resources and media that are intentionally constructed, shared, manipulated and used by agents to support their activities, either cooperatively or competitively
- *Workspaces*: as conceptual containers of agents and artifacts, useful for defining the topology for the environment and providing a way to define a notion of locality

Figure 1 shows a concept of the proposed model. The goal-based agent interacts with the surrounded environment as known as workspace by using the sensors. However, the proposed model does not actively interact with artifacts. For example, an ant consumes pray to prolong its life, but the model does not consider the '*consume*' behavior as an interaction. Instead, it is considered as a process that the energy of the pray was transferred to the ant.

The following subsections will describe each component of the model. We aim at a multi-robot environment in which robots are the same in terms of its mission and functionality. However, a goal is an atomic entity such that it cannot be shared or combined with other goals.

3.1 Goal-Based Modeling

Goal-based model is one of widely used modeling techniques in engineering [27–29] and standardizes goals for robots. Goal-based model consists of a set of sub-goals, which are the entities to achieve the super goal. Behaviors of a robot are determined based on what robot has to do at the moment to finish the

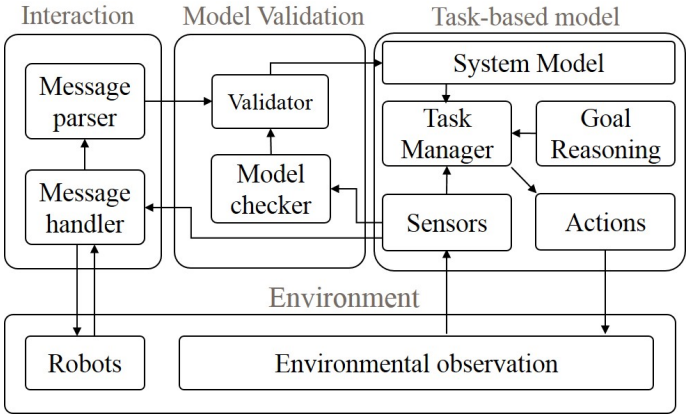

Figure 1. A diagram of the proposed model.

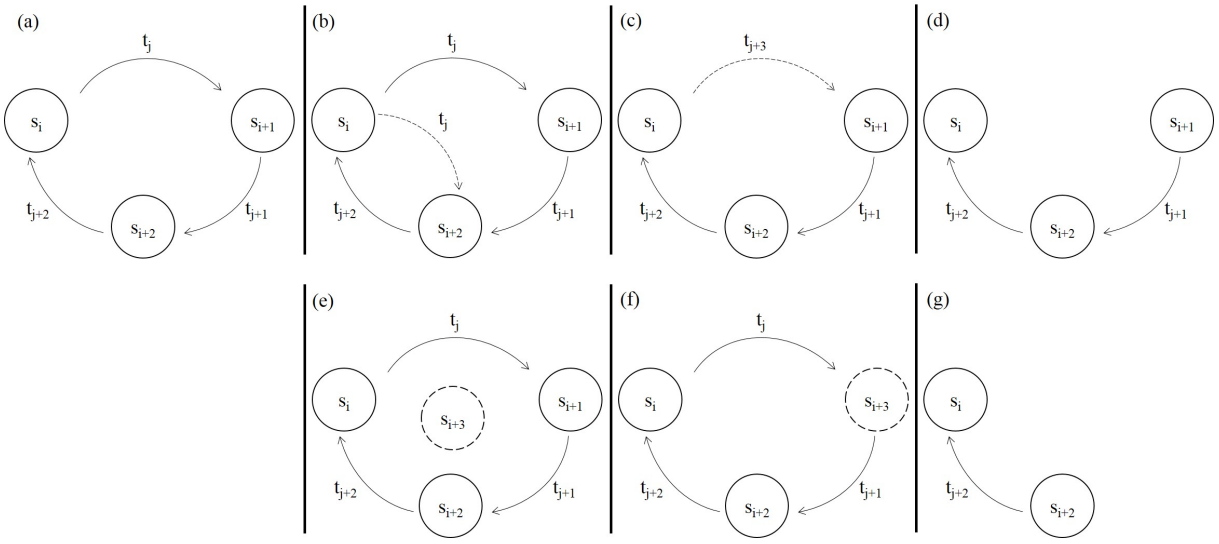

Figure 2. Examples of changes in the goal-based model. (a) An original model. (b) Add, (c) modify, and (d) remove a transition. (e) Add, (f) modify, and (g) remove a state from the original model. For (g), transitions that were connected to the state are also removed.

current task that satisfies the given goal. Advantages of goal-based model are 1) simple enough to design, 2) quantitatively analyzable (i.e., its outcome can be measured), and 3) possible to logically represent states. In particular, the advantage of logical expression of states is useful to check validity of the system.

A goal-based model is a tuple $Robot = \{S, T, I, \rightarrow, AP\}$, where,

- S is a set of states;
- T is a set of transitions;
- $I = s_0 \in S$ is an initial state;
- $\rightarrow \subseteq S \times T \times S$ is a transition; and
- *AP* is a set of atomic propositions.

A state $s_i \in S$ represents a goal. We assume that each state is unique such that only transitions $t_k \in \{s_i \in S,\ s_j \in S,\ s_i \xrightarrow{t_k} s_j | i \neq j\}$, for $0 < k <$ *number of transitions*, are considered. Let *an infinite path fragment* be $\pi = s_0, s_1, ..., s_n$, where n is infinite. Let *a finite path fragment* be $\widehat{\pi} = s_0, s_1, ..., s_m$, where m is finite. What the model checker is looking for is a path fragment that explains whether a goal state is reachable from the current state.

The set T is triggered in many ways of interactions with environment: performing an action by the robot, performing an action by other robots, or an environmental change. The second way can be made by exchanging information among robots. *AP* is used to convert received message from other robot into a corresponding trigger that makes a change in the model. For example, a ground robot found that the end of the road is closed by an obstacle while other ground robots are still driving toward the end of the road. The ground robot that found the obstacle sends a message to other ground robots in order to let them know. And then, the message is translated based on message sets defined in *AP* and triggers a transition in the other ground robots such that they change to look for detour.

This component is a typical behavioral model that contains all functions and reasoning process for a robot to perform tasks. Before deployment of a robot, we assume that system model and functions are working correctly. We do not consider how well a robot performs goal reasoning because it is out of scope of this research.

3.2 Adaptive Model Validation

In order for autonomous robots to address dynamic environment, their model has to be dynamically changeable. As shown in Figure 2, states and transitions are subject to be changed in goal-based model. Let *robot* be the goal model and let *robot′* be the model after a change is made. There are three types of operations to make a change in *robot′*: *add*, *modify*, and *remove*. We are actually not interested in adding a state or transition because *add* operation is likely to be less influential than other operations in terms of attainment of the goals. This does not mean that we do not need to check the system after *add* operation is executed; we need to check to find *a finite path*

fragment to keep satisfying system requirement. Followings are processes for the rest four cases that we are interested in,

- **Modify a state**: A state is changed. However, transitions with its predecessors and successors remain to make a transition with the changed state. In this case, the robots functionality is not changed but, purpose of the task is changed.
- **Remove a state**: We assume current state at the moment is not the same with the removing state. When the state is removed, corresponding predecessors and successors are also removed. The robot disables corresponding functionality to the transitions that were removed.
- **Modify a transition**: Even though this change does not much alter the model, the transition that represents robots functionality is changed.
- **Remove a transition**: In this case, robot disables or loses one capability. This change is critical because there is a possibility that a finite path fragment from the current state to the goal state does not exist.

As described in Section 2, model checking technique is a tool that checks a system based on the given specification. Model checker gives us not only result of satisfaction of the given specification, but also tells us a counterexample if the result is 'FALSE.' In order to verify that robots can continue working after a change is occurred, we need to define two specifications. The two specifications say '*can the robot accomplish current task and go to the next task*?', '*can the robot achieve the goal by accomplishing corresponding tasks*?'. The two specifications are respectively expressed in LTL form as follows,

$$\exists(s_{current}) \rightarrow X(s_{current+1}) \tag{1}$$

$$\exists(s_{current} \rightarrow \exists s_{goal}) \tag{2}$$

NuSMV is one of Binary Decision Diagram (BDD)-based symbolic model checkers. NuSMV uses text-based description of a model and analyzes it based on the given specification. It produces a counterexample of the specification if the result is 'FALSE.' The counterexample can be utilized to analyze robots status after robot senses a stimulus that changes the model. Figure 3 shows an example of counterexamples from the case where the model is changed from Figure 2(a) to 2(d). In this example, the robot lost a capability so that it is no longer for it

(a)

```
-- specification ( F states = s1 ->  X states = s2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  states = s0
-> State: 1.2 <-
  states = s1
-- Loop starts here
-> State: 1.3 <-
  states = s2
-> State: 1.4 <-
```

(b)

```
-- specification ( F states = s1 ->  F states = s3)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  states = s0
-> State: 1.2 <-
  states = s1
-- Loop starts here
-> State: 1.3 <-
  states = s2
-> State: 1.4 <-
```

Figure 3. Counterexamples of the case where original model is changed from Figure 2(a) to 2(d). (a) and (b) are counterexamples of the specification (1) and (2), respectively.

to finish the current task and go to either next or goal task. In the practical implementation, even though proposed model allows a change of state, we focus only on changing transitions due to the fact that modifying a state could change entire model (e.g., remove corresponding transitions) that results in invalidation of the model or the model is no longer valid to accomplish the given goal. In addition, we assume that a transition that does not exist in an original model at design phase cannot be added. We will address this assumption when we consider capability changeable robot, which adds H/W or S/W type capability in run-time to expend its functionality [30].

3.3 Dynamic Message Exchanging

The proposed model accepts messages from other robots in the same workspace to apply any changes to the model. The HARMS model is a tool that provides organizational communications and interactions between agents. The HARMS model utilizes network-based communication, such which agents exchange information through sending messages over a network. The messages are purpose-specific and some message types require feedback from the receiver. For example, when agent *A* sends a message to command agent *B*, agent *B* will reply with a message to agent *A*.

HARMS model provides three fundamental message types (i.e., *Notification*, *Query*, and *Command*) to allow robots to exchange information using the messages types. We use notification-type message with multi-cast transmission in order to send a message to nearby neighbor robots. The message parser in

Table 1. An example of *AP* sub-sets and corresponding Human-Agent-Robot-Machine-Sensor (HARMS) messages

AP=$\{s_0,s_1,s_2,t_0,t_1,t_2\}$	HARMS message
AP_{add}=$\{s_1,t_1,t_2\}$	"Either Action 1 or Action 2 can accomplish Goal 1"
AP_{mod}=$\{s_1,t_2\}$	"Goal 1 can be accomplished by only Action 2"
AP_{rem}=$\{s_1\}$	"Goal 1 cannot be accomplished by any actions"

the model translates received message to corresponding *AP* in the goal-based model. Table 1 shows an example of look-up table for the example illustrated in Figure 2.

4. Conclusion

Autonomous robots in dynamic environment should be able to continue their mission to maintain reliability of the system. Because unwanted changes from the environment may disrupt the robots while working, the robots have to detect and handle such changes. The proposed model is an enhanced model from the classic goal-based model, which uses HARMS and the model checker in order to detect and handle the changes. The most significant advantage of using the proposed model is automated run-time validation.

As mentioned in Section 2, attempts of applying model checking technique to the field of robotics are very active in the fields [22–25, 31, 32]. However, the model checking tools that were used in the studies vary and run standalone. This means that it is difficult to merge it with the simulation tools for multi-agent systems such as robot operating system (ROS) [33] and Netlogo [34].

The proposed model can fully demonstrate its capability in scenarios where autonomous robots have limited information to proceed their mission. As an example of applications, Zhong and DeLoach proposed a goal-based robot system [35] that has a capability to reorganize the robots themselves. In the scenario the robots have roles assigned based on their capability to the role and make an organization for the mission. When some of the robots stopped working, implying an environmental change to other robots, the remaining robots start reorganizing by re-assigning new roles to the remainders for the mission. This approach works only if there is a robot that can substitutes the stopped one, which means that the substituted robot has the capability to take the role from the stopped robot. Because the proposed model tries to find an alternative for robot to continue proceeding the goal, rather than being stopped, applying model checking technique to such scenario provides more reliability and flexibility to accomplish mission.

This research study is actively on progress through the published work and this extension. Our next work would be to simplify the existing model checker to make the checker embed into one of the simulation tools. In order to validate our work, we will try to apply the model checker to Netlogo, especially into the ant model built by Wilensky [36]. The ant model is the most simple and appropriate environment to prove concept of the proposed model.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Number CNS-1239171 and CNS-1239196. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

This work was partially supported by the Basic Science Research Program (2015R1D1A1A09061368) through the National Research Foundation of Korea funded by the Korean government (MEST).

References

[1] Q. Gao and Y. I. Cho, "A dynamic ontology-based multi-agent context-awareness user profile construction method for personalized information retrieval," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 12, no. 4, pp. 270-276, 2012. http://dx.doi.org/10.5391/IJFIS.2012.12.4.270

[2] E. G. Kim, "An approach to generate a theory of coordination for multi-agent systems," *International Journal of*

Fuzzy Logic and Intelligent Systems, vol. 4, no. 3, pp. 277-282, 2004. http://dx.doi.org/10.5391/IJFIS.2004.4.3.277

[3] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, and M. Barreto, "Ubiquitous robotics: recent challenges and future trends," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1162-1172, 2013. http://dx.doi.org/10.1016/j.robot.2013.04.003

[4] S. A. Deloach, W. H. Oyenan, and E. T. Matson, "A capabilities-based model for adaptive organizations," *Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 1, pp. 13-56, 2008. http://dx.doi.org/10.1007/s10458-007-9019-4

[5] J. W. Park, Y. S. Son, J. W. Jung, and S. M. Oh, "R-object model for evolutionary robots using multi-robot cooperation," *IFAC Proceedings Volumes: International Federation of Automatic Control*, vol. 42, no. 19, pp. 438-443, 2009. http://dx.doi.org/10.3182/20090921-3-TR-3005.00077

[6] H. H. Viet, S. H. An, and T. C. Chung, "Univector field method based multi-agent navigation for pursuit problem," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 12, no. 1, pp. 86-93, 2012. http://dx.doi.org/10.5391/IJFIS.2012.12.1.86

[7] E. T. Matson and B. C. Min, "M2M infrastructure to integrate humans, agents and robots into collectives," in *Proceedings of 2011 IEEE Instrumentation and Measurement Technology Conference*, Binjiang, China, 2011, pp. 1-6. http://dx.doi.org/10.1109/IMTC.2011.5944359

[8] E. T. Matson, J. Taylor, V. Raskin, B. C. Min, and E. C. Wilson, "A natural language exchange model for enabling human, agent, robot and machine interaction," in *Proceedings of 2011 5th International Conference on Automation, Robotics and Applications*, Wellington, New Zealand, 2011, pp. 340-345. http://dx.doi.org/10.1109/ICARA.2011.6144906

[9] Y. Kim, J. W. Jung, and E. T. Matson, "An adaptive task-based model for autonomous multi-robot using HARMS and NuSMV," *Procedia Computer Science*, vol. 56, pp. 127-132, 2015. http://dx.doi.org/10.1016/j.procs.2015.07.180

[10] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285-312, 2000. http://dx.doi.org/10.1023/A:1010071910869

[11] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 432-456, 2008. http://dx.doi.org/10.1007/s10458-008-9053-x

[12] S. Rodriguez, V. Hilaire, N. Gaud, S. Galland, and A. Koukam, "Holonic multi-agent systems," in *Self-organising Software*, G. Di Marzo Serugendo, M. P. Gleizes, and A. Karageorgos, Eds. Berlin: Springer-Verlag, 2011, pp. 251-279. http://dx.doi.org/10.1007/978-3-642-17348-6_11

[13] O. Boissier, R. H. Bordini, J. F. Hubner, A. Ricci, and A. Santi, "Multi-agent oriented programming with JaCaMo," *Science of Computer Programming*, vol. 78, no. 6, pp. 747-761, 2013. http://dx.doi.org/10.1016/j.scico.2011.10.004

[14] S. Rodriguez, N. Gaud, and S. Galland, "SARL: a general-purpose agent-oriented programming language," in *Proceedings of 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, Warsaw, Poland, 2014, pp. 103-110. http://dx.doi.org/10.1109/WI-IAT.2014.156

[15] N. R. Jennings, "Agent-oriented software engineering," in *Multiple approaches to intelligent systems,* I. Imam, Y. Kodratoff, A. El-Dessouki, and M. Ali, Eds. Berlin: Springer-Verlag, 1999, pp. 4-10. http://dx.doi.org/10.1007/978-3-540-48765-4_2

[16] A. Ricci and A. Santi, "Agent-oriented computing: agents as a paradigm for computer programming and software development," in *Proceedings of the Third International Conference on Future Computational Technologies and Applications*, Rome, Italy, 2011, pp. 42-51.

[17] M. Wooldridgey and P. Ciancarini, "Agent-oriented software engineering: the state of the art," in *Agent-oriented software engineering*, P. Ciancarini and M. J. Wooldridge, Eds. Berlin: Springer-Verlag, 2001, pp. 1-28. http://dx.doi.org/10.1007/3-540-44564-1_1

[18] S. Rodriguez, N. Gaud, V. Hilaire, S. Galland, and A. Koukam, "An analysis and design concept for self-organization in holonic multi-agent systems," in *Engineering Self-Organising Systems*, S. A. Brueckner, S. Has-

sas, M. Jelasity, and D. Yamins, Eds. Berlin: Springer-Verlag, 2007, pp. 15-27. http://dx.doi.org/10.1007/978-3-540-69868-5_2

[19] C. Baier and J. P. Katoen, *Principles of model checking*. Cambridge, MA: MIT Press Cambridge, 2008.

[20] A. Chiappini, A. Cimatti, L. Macchi, O. Rebollo, M. Roveri, A. Susi, S. Tonetta, and B. Vittorini, "Formalization and validation of a subset of the European Train Control System," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, Cape Town, South Africa, 2010, pp. 109-118. http://dx.doi.org/10.1145/1810295.1810312

[21] C. S. Pasareanu, P. C. Mehlitz, D. H. Bushnell, K. Gundy-Burlet, M. Lowry, S. Person, and M. Pape, "Combining unit-level symbolic execution and system-level concrete execution for testing nasa software," in *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, Seattle, WA, 2008, pp. 15-26. http://dx.doi.org/10.1145/1390630.1390635

[22] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation,* Barcelona, Spain, 2005, pp. 2020-2025. http://dx.doi.org/10.1109/ROBOT.2005.1570410

[23] S. Konur, C. Dixon, and M. Fisher, "Analysing robot swarm behaviour via probabilistic model checking," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 199-213, 2012. http://dx.doi.org/10.1016/j.robot.2011.10.005

[24] J. Goppert, J. C. Gallagher, I. Hwang, and E. Matson, "Model checking of a flapping-wing mirco-air-vehicle trajectory tracking controller subject to disturbances," in *Robot Intelligence Technology and Applications 2*, J. H. Kim, E. T. Matson, H. Myung, P. Xu, and F. Karray, Eds. Cham: Springer International Publishing, 2014, pp. 531-543. http://dx.doi.org/10.1007/978-3-319-05582-4_46

[25] Y. Kim, M. Gomez, J. Goppert, and E. T. Matson, "Model checking of a training system using nusmv for humanoid robot soccer," in *Robot Intelligence Technology and Applications 3*, J. H. Kim, W. Yang, J. Jo, P. Sincak, and H. Myung, Eds. Cham: Springer International Publishing, 2015, pp. 531-540. http://dx.doi.org/10.1007/978-3-319-16841-8_48

[26] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV 2: an opensource tool for symbolic model checking," in *Computer Aided Verification*, E. Brinksma and K.G. Larsen, Eds. Berlin: Springer-Verlag, 2002, pp. 359-364. http://dx.doi.org/10.1007/3-540-45657-0_29

[27] J. J. Shi, D. E. Lee, and E. Kuruku, "Task-based modeling method for construction business process modeling and automation," *Automation in Construction*, vol. 17, no. 5, pp. 633-640, 2008. http://dx.doi.org/10.1016/j.autcon.2007.10.010

[28] S. A. DeLoach and M. Miller, "A goal model for adaptive complex systems," *International Journal of Computational Intelligence: Theory and Practice*, vol. 5, no. 2, pp. 83-92, 2010.

[29] M. A. Goodrich and D. Yi, "Toward task-based mental models of human-robot teaming: a Bayesian approach," in *Virtual augmented and mixed reality: designing and developing augmented and virtual environments*, R. Shumaker, Ed. Berlin: Springer-Verlag, 2013, pp. 267-276. http://dx.doi.org/10.1007/978-3-642-39405-8_30

[30] Y. S. Son, Y. S. Lee, and J. W. Jung, "Software modules management techiques for multi-cooperate robots based on r-object model in dynamic environments," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 7, no. 4, pp. 163-174, 2012.

[31] W. Visser, K. Havelund, G. Brat, S. J. Park, and F. Lerda, "Model checking programs," *Automated Software Engineering*, vol. 10, no. 2, pp. 203-232, 2003. http://dx.doi.org/10.1023/A:1022920129859

[32] Lerda, J. Kapinski, H. Maka, E. M. Clarke, B. H. Krogh, "Model checking in-the-loop: finding counterexamples by systematic simulation," in *Proceedings of 2008 American Control Conference*, Seattle, WA, 2008, 2734-2740. http://dx.doi.org/10.1109/ACC.2008.4586906

[33] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *Proceedings of ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.

[34] S. Tisue and U. Wilensky, "NetLogo: Design and implementation of a multi-agent modeling environment," in

Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence, Chicago, IL, 2004.

[35] C. Zhong and S. A. DeLoach, "Runtime models for automatic reorganization of multi-robot systems," in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Honolulu, Hawaii, 2011, pp. 20-29. http://dx.doi.org/10.1145/1988008.1988012

[36] U. Wilensky, "Netlogo Ants model," Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997. Available http://ccl.northwestern.edu/netlogo/models/Ants

Yongho Kim is a PhD candidate in M2M laboratory at Purdue University. He received B.S degree in Computer Science and Engineering at Seoul National University of Science and Technology, Korea. He also received M.S. degree in Electrical and Radio Engineering at Kyung Hee University, Korea. His interests are multi-agent robots, communications and interactions between heterogeneous agents, and behavior verification of multi agent organizations.

Jin-Woo Jung received the B.S. and M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1997 and 1999, respectively and received the Ph.D. degree in electrical engineering and computer science from KAIST, Korea in 2004. Since 2006, he has been with the Department of Computer Science and Engineering at Dongguk University, where he is currently an Associate Professor. During 2001-2002, he worked as visiting researcher at the Department of Mechano-Informatics, University of Tokyo, Japan. During 2004-2006, he worked as researcher in Human-friendly Welfare Robot System Research Center at KAIST, Korea. During 2014, he worked as visiting scholar at Purdue University, USA. His current research interests include human behavior recognition with emphasis on biometric application, multiple robot cooperation and intelligent human-robot interaction.

John C. Gallagher is a Professor in the Department of Computer Engineering and Science and in the Department of Electrical Engineering at Wright State University. He has research interests in deep learning, neuromorphic computation, metaheuristc search, cyberphysical systems, robotics, and engineering education. He has a Ph.D. in Computer Engineering, a M.S. in Computer Science, and a B.S. in Computer Engineering from Case Western Reserve University in Clevelenad, Ohio.

Eric T. Matson is an Associate Professor in the Department of Computer and Information Technology at Purdue University (West Lafayette). Prof. Matson was a Visiting Professor in the Department of Computer Science and Engineering at Dongguk University, Seoul, Korea and was an International Faculty Scholar, Department of Radio and Electronics Engineering, College of Electronics and Information, Kyung Hee University, Suwon, Korea. He co-founded the M2M Lab at Purdue University, which performs research in multiagent systems, cooperative robotics, and wireless communication. He is also the Founder and Director of the Center for Robotic Innovation, Commercialization and Education (RICE) at Purdue University. Prior to his position at Purdue University, Prof. Matson was in industrial and commercial software development as a consultant, software engineer, manager and director for 14 years. In that experience, he developed and led numerous large software engineering projects dealing with intelligent systems, applied artificial intelligence and distributed object technologies. Prof. Matson has a Ph.D. in Computer Science and Engineering from the University of Cincinnati, M.B.A in Operations Management from Ohio State University, M.S.E. in Software Engineering from Kansas State University and a B.S. in Computer Science from Kansas State University.