

# ULRR

## Knowledge representation for self-adaptive behavior

Item Type	Meetings and Proceedings
Authors	Vassev, Emil;Hinchey, Mike;Gaudin, Benoit
Citation	C3S2E'12 Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering;pp. 113-117
Publisher	Association for Computing Machinery
Download date	2026-06-13 02:32:22
Item License	<a href="https://creativecommons.org/licenses/by-nc-sa/1.0/">https://creativecommons.org/licenses/by-nc-sa/1.0/</a>
Link to Item	<a href="https://hdl.handle.net/10344/2592">https://hdl.handle.net/10344/2592</a>

# Knowledge Representation for Self-Adaptive Behavior

Emil Vassev

Lero – the Irish Software Engineering  
Research Centre,  
University of Limerick,  
Limerick, Ireland  
emil.vassev@lero.ie

Mike Hinchey

Lero – the Irish Software Engineering  
Research Centre,  
University of Limerick,  
Limerick, Ireland  
mike.hinchey@lero.ie

Benoit Gaudin

Lero – the Irish Software Engineering  
Research Centre,  
University of Limerick,  
Limerick, Ireland  
benoit.gaudin@lero.ie

## ABSTRACT

An autonomic system is considered to be a self-adaptive system that changes its behavior in response to stimuli from its execution and operational environment. Such behavior is considered autonomic and self-adaptive and is intended to drive intelligent systems in situations requiring adaptation. Such systems encapsulate rules, constraints and mechanisms for self-adaptation and acquire and process knowledge about themselves and their environment. In this paper, an approach to knowledge representation and reasoning for self-adaptive behavior is presented. The approach is formal and demonstrates how knowledge representation and reasoning help to establish the vital connection between knowledge, perception, and actions realizing the self-adaptive behavior. The knowledge is used against the perception of the world to generate appropriate actions in compliance to some goals and beliefs.

## Categories and Subject Descriptors

I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods – Representation languages; D.3.2 [Programming Languages]: Language Classifications – Very high-level languages;

## General Terms

Algorithms, Design, Experimentation, Languages, Performance

## Keywords

knowledge representation, reasoning, self-adaptation

## 1. INTRODUCTION

A part from the complex mechanisms and electronics, building intelligent systems is about the challenge of interfacing with a dynamic and unpredictable world, which requires presence of intelligence. Today, artificial intelligence (AI) mainly excels at formal logic, which allows it, for example, to find the right chess move from hundreds of previous games. Intelligent system engineers use knowledge representation techniques to give computerized systems large amounts of knowledge that helps

them understand the problem domain and solve complex problems. Knowledge representation primitives such as *rules*, *frames*, *semantic networks*, *concept maps*, *ontologies*, and *logic expressions* might be used to represent distinct pieces of knowledge that are worth being differently represented. Moreover, these primitives might be combined into more complex knowledge elements. Whatever elements they use, engineers must structure the knowledge so that the system can effectively process it and eventually derive its own behavior.

Decision-making is a complex process that is often based on more than *logical conclusions*. Probability and statistics may provide for the so-called *probabilistic* and *statistical reasoning* intended to capture uncertain knowledge in which additive probabilities are used to represent degrees of belief of rational agents in the truth of statements. For example, the purpose of a statistical inference might be to draw conclusions about a population based on data obtained from a sample of that population. Probability theory and Baye's theorem [1] lay the basis for such reasoning where Bayesian networks [2] are used to represent belief probability distributions, which actually summarize a potentially infinite set of possible circumstances. The key point is that nodes in a Bayesian network have direct influence on other nodes and given values for some nodes, it is possible to infer the probability distribution for values of other nodes. How a node influences another node is defined by the conditional probability for the nodes usually based on past experience. The experience can be associated with the success of the actions generated in the physical environment by the intelligent system. Maintaining an execution history of the actions shall help that system eventually compute the success probability for those actions. In that way, the system may learn (infer new knowledge) not to execute actions that traditionally have low success rate.

In this paper, we present how we use this methodology to build a *knowledge representation model for self-adaptive behavior*.

The rest of this paper is organized as follows. Section 2 justifies the necessity of knowledge representation and briefly presents KnowLang, a formal specification language for knowledge representation in self-adaptive systems. Section 3 presents our approach to knowledge representation for self-adaptive behavior. In Section 4, we present a proof-of-concept case study and finally, Section 5 presents a brief conclusion and future work.

## 2. KNOWLEDGE AND KNOWLEDGE REPRESENTATION

When it comes to AI, we think about the knowledge we must transfer to the computerized machines and make them use that

knowledge, so they become intelligent. In this regard, one of the first questions we need to answer is on the notion of knowledge. So, what is knowledge? To answer this question we should consider two facts: 1) it is known that knowledge is related to intelligence; and 2) the definition of knowledge should be given with terms from the computer domain. Scientists agree that the concept of intelligence is built upon four fundamental elements: data, information, knowledge, and wisdom. In general, data takes the form of measures and representations of the world—for example, raw facts and numbers. Information is obtained from data by assigning relevant meaning, usually by putting data in a specific context. Knowledge is a specific interpretation of information. And wisdom is the ability to apply relevant knowledge to a particular problem.

### 2.1 Why Knowledge Representation?

Intelligent system designers use knowledge representation to give computerized systems large amounts of knowledge that helps them understand the problem domain. Still computers “talk” in a “binary” language, which is simple, logical, and sound, and has no sense of ambiguity typical for a human language. Therefore, computers cannot be simply given textbooks, which they understand and use, just like human do. Instead, the knowledge given to computers must be structured in well-founded computational structures that computer programs may translate to the binary computer language. Knowledge representation structures could be primitives such as *rules*, *frames*, *semantic networks* and *concept maps*, *ontologies*, and *logic expressions*. These primitives might be combined into more complex knowledge elements. Whatever elements they use, designers must structure the knowledge so that the system can effectively process and it and humans can easily perceive the results.

Many conventional developers doubt the utility of knowledge representation (KR). Fact is that knowledge representation and the accompanying reasoning can significantly slow a system down when it has to decide what actions to take, and it looks up facts in a knowledge base to reason with them at runtime. This is one of the main arguments against knowledge representation. Why not simply “compile out” the entire knowledge as “procedural knowledge”, which makes the system relatively faster and more efficient. However, this strategy will work for a fixed set of tasks, i.e., procedural knowledge will give the system the entire knowledge the system needs to know. However, AI deals with an open set of tasks and those cannot be determined in advance (at least not all of them). This is the big advantage of using knowledge representation – AI needs it to solve complex problems where the operational environment is non-deterministic and a system needs to reason at runtime to find missing answers.

### 2.2 Knowledge Representation for ASCENS

Autonomic Service-Component ENsembles (ASCENS) [3] is an FP7 (Seventh Framework Program) [4] project targeting the development of a coherent and integrated set of methods and tools providing a comprehensive development approach to developing *ensembles* (or *swarms*) of intelligent, self-aware and adaptive *service components*. One of the main scientific contributions that we expect to achieve with ASCENS is related to *knowledge representation and reasoning* (KR&R.). Note that it is of major importance for an ASCENS system to acquire and structure comprehensive knowledge in such a way that it can be effectively and efficiently processed, so such a system becomes aware of itself and its environment. Moreover, ASCENS is an AI project

tackling self-adaptation of systems operating in open-ended environment, e.g., our physical world. Such systems need to be developed with initial knowledge and learning capabilities based on knowledge processing and awareness. It is very important how the system knowledge is both structured and modeled to provide essence of awareness and self-adaptation.

KnowLang [5, 6] is an initiative undertaken by Lero – the Irish Software Engineering Research Center within Lero’s mandate in the ASCENS project. A key feature of KnowLang is a multi-tier specification model allowing for integration of ontologies together with rules and Bayesian networks [5]. The language aims at efficient and comprehensive knowledge structuring and awareness based on logical and statistical reasoning. It helps us tackle 1) explicit representation of domain concepts and relationships; 2) explicit representation of particular and general factual knowledge, in terms of predicates, names, connectives, quantifiers and identity; and 3) uncertain knowledge in which additive probabilities are used to represent degrees of belief. Other remarkable features are related to knowledge cleaning (allowing for efficient reasoning) and knowledge representation for autonomic robotic behavior.

### 2.3 KnowLang

KnowLang imposes a multi-tier specification model (see Figure 1), where we specify *knowledge corpuses*, *KB (knowledge base) operators* and *inference primitives* at different hierarchically organized tiers. As shown in Figure 1, knowledge is organized in a special Knowledge Base (KB) at three main tiers: 1) Knowledge Corpuses; 2) KB Operators; and 3) Inference Primitives.

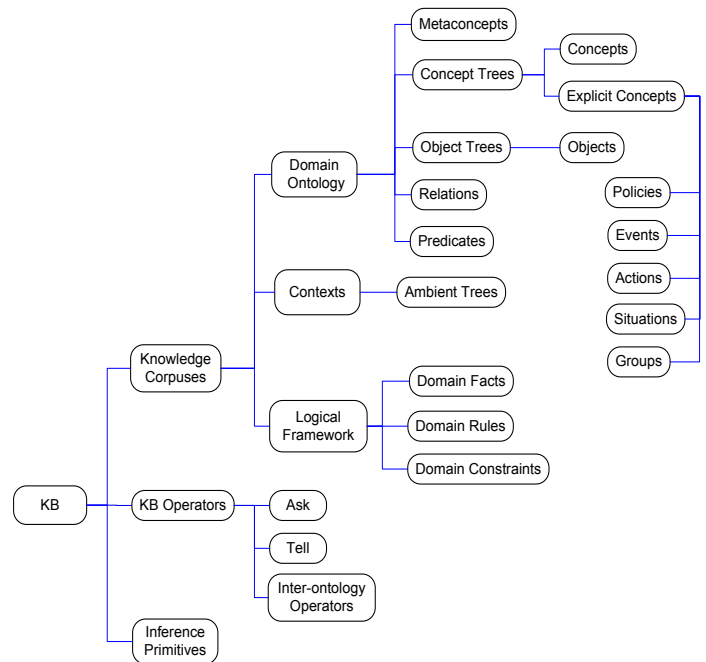


Figure 1. KnowLang Multi-tier Specification Model [5]

The tier of Knowledge Corpuses is used to specify KR structures. The tier of KB Operators provide access to Knowledge Corpuses via special class of ASK and TELL operators where ASK operators are dedicated to knowledge querying and retrieval and TELL operators allow for knowledge update. Moreover, this tier

provides for special *inter-ontology operators* intended to work on one or more ontologies. Note that all the KB Operators may imply the use of *Inference Primitives*, i.e., new knowledge might be inferred and eventually stored in the KB. The tier of Inference Primitives is intended to specify algorithms for reasoning and knowledge inference.

In this paper, we do not present the language itself, but the embedded KR mechanism for representing self-adaptive behavior. The interested reader is advised to refer to [5] for more information on the KnowLang's specification model.

### 3. KNOWLEDGE REPRESENTATION FOR SELF-ADAPTIVE BEHAVIOR

An *autonomic system* [7] is considered to be a self-adaptive system that changes its behavior in response to stimuli from its execution and operational environment. Such behavior is considered *autonomic* and *self-adaptive* [7] and is intended to drive a system in situations requiring adaptation. Any long-running system is subject to uncertainty in its execution environment due to potential changes in requirements, business conditions, available technology, etc. Thus, it is important to capture and cater for uncertainty as part of the development process. Failure to do so may result in systems that are too rigid to be fit for purpose, which is of particular concern for the domains that typically make use of self-adaptive technology, e.g., ASCENS. We hypothesize that modeling uncertainty and developing mechanisms for managing it as part of Knowledge Representation & Reasoning (KR&R) will lead to systems that are:

- more expressive of the real world;

$\Pi := \{\pi_0, \pi_1, \dots, \pi_n\}$	(Policies)	(1)
$\pi := \{g, Si_\pi, [R_\pi], N_\pi, A_\pi, \text{map}(N_\pi, A_\pi)\}$	(Policy)	(2)
$A_\pi \subset A_{si}, N_\pi \rightarrow A_\pi$	( $A_\pi$ – Policy Actions)	
$E_\pi \subset E, E_\pi \subset N_\pi$	( $E_\pi$ – Policy Events)	
$Si_\pi \subset Si, Si_\pi \rightarrow \pi \rightarrow N_\pi$	( $Si_\pi$ – Policy Situations)	
$R_\pi \subset R, r_\pi := \{\pi, [rn], [Z], si_\pi\}$	( $R_\pi$ – Policy – Situation Realtion )	
$N_\pi := \{n_0, n_1, \dots, n_n\}$	(Policy Conditions)	(3)
$n := bf(O) \mid E_\pi \mid Si_\pi$	(Condition – Boolean Statement, Ev., Situations)	(4)
	(O – Ontology defines the system's operational domain)	
$g := (s \Rightarrow s')$	(Goal)	(5)
$Si := \{si_0, si_1, \dots, si_n\}$	(Situations)	(6)
$si := \{s, A_{si}^{\leftarrow}, [E_{si}^{\leftarrow}], A_{si}\}$	(Situation)	(7)
$A_{si}^{\leftarrow} \subset A$	( $A_{si}^{\leftarrow}$ – Executed Actions)	
$A_{si} \subset A$	( $A_{si}$ – Possible Actions)	
$E_{si}^{\leftarrow} \subset E$	( $E_{si}^{\leftarrow}$ – Situation Events )	

Ideally, policies are specified to handle specific situations, which may trigger the application of policies. A policy exhibits a behavior via actions generated in the environment or in the system itself. Specific conditions determine, which specific actions (among the actions associated with that policy – see Definition 2) shall be executed. These conditions are often generic and may differ from the situations triggering the policy. Thus, the behavior not only depends on the specific situations a policy is specified to handle, but also depends on additional conditions. Such conditions might be organized in a way allowing for synchronization of different situations on the same policy. When a policy is applied,

- fault tolerant due to fluctuations in requirements and conditions being anticipated;
- flexible and able to manage dynamic changes.

### 3.1 Formal Approach

The ability to represent knowledge providing for self-adaptive behavior is an important factor in dealing with uncertainty. In our approach, the *autonomic self-adapting behavior* is provided by *policies, events, actions, situations, and relations* between policies and situations (see Definitions 1 through 8). In our KR&R model Policies ( $\Pi$ ) are responsible for the *autonomic behavior*. A policy  $\pi$  has a *goal* ( $g$ ), *policy situations* ( $Si_\pi$ ), *policy-situation relations* ( $R_\pi$ ), and *policy conditions* ( $N_\pi$ ) mapped to *policy actions* ( $A_\pi$ ), where the evaluation of  $N_\pi$  may imply the evaluation of actions (denoted with  $N_\pi \rightarrow A_\pi$ ) (see Definition 2). A *condition* is a Boolean function over ontology (see Definition 4) or the occurrence of specific events or situations in the system. Thus, policy conditions may be expressed with *policy events*. Policy situations ( $Si_\pi$ ) are situations (see Definition 6) that may trigger a policy  $\pi$ , which implies the evaluation of the policy conditions  $N_\pi$  (denoted with  $Si_\pi \rightarrow \pi \rightarrow N_\pi$ ). A policy may also comprise optional *policy-situation relations* ( $R_\pi$ ) justifying the relationships between a policy and the associated situations. The presence of probabilistic belief in those relations justifies the probability of policy execution, which may vary with time. A *goal* is a desirable transition from a *state* to another *state* (denoted with  $s \Rightarrow s'$ ) (see Definition 5). A *situation* is expressed with a state ( $s$ ), a *history of actions* ( $A_{si}^{\leftarrow}$ ) (actions executed to get to state  $s$ ), *actions*  $A_{si}$  that can be performed from state  $s$  and an optional *history of events*  $E_{si}^{\leftarrow}$  that eventually occurred to get to state  $s$  (see Definition 7).

it checks what particular conditions are met and performs the associated actions (see  $\text{map}(N_\pi, A_\pi)$  – see Definition 2).

The cardinality for the *policy-event relationship* is many-to-many, i.e., a situation might be associated with many policies and vice versa. Moreover, the set of *policy situations* (situations triggering a policy) is open-ended, i.e., new situations might be added or old might be removed from there by the system itself. Moreover, with a set of *policy-situation relations* we may grant the system with an initial *probabilistic belief* (see Definition 2) that certain situations require specific policies to be applied. Runtime factors may change this probabilistic belief with time, so the most likely

situations a policy is associated with can be changed. For example, the successful rate of actions execution associated with a specific situation and a policy may change such a probabilistic belief and place a specific policy higher in the “list” of associated policies, which will change the behavior of the system when a specific situation is to be handled. Note that situations are associated with a state (see Definition 7) and a policy has a goal (see Definition 2), which is considered as a transition from one state to another (see Definition 5). Hence, the *policy-situation relations* and the employed *probabilistic beliefs* may help a cognitive system what desired state to choose, based on past experience.

#### 4. CASE STUDY

As a proof of concept, we applied our approach to one of the ASCENS case studies – the ensemble of robots case study [3]. This case study targets swarms of intelligent individual robots with self-awareness capabilities that help the entire swarm acquire the capacity to reason, plan and autonomously act. This shall give the robot swarm self-adapting capabilities and more goal-oriented and efficient use of resources. The case study is based on the marXbot robotic platform [8].

##### 4.1 The marXbot Robotic Platform

The marXbot [8] is a modular research robot equipped with a set of devices that help the robot interact with other robots or the robotic environment. The environment is defined as an arena where special cuboid-shaped obstacles are present in arbitrary positions and orientations. Moreover, the environment may contain a number of light sources, usually placed behind the goal area, which act as environmental cues used as shared reference frames among all robots.

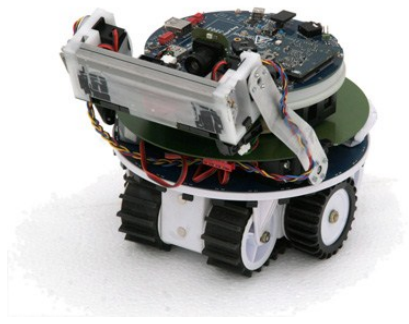


Figure 2. A marXbot Robot [8]

Figure 2 shows a marXbot robot [8]. Such robot is equipped with a set of devices to interact with the environment and with other robots of the swarm:

- a light sensor, that is able to perceive a noisy light gradient around the robot in the 2D plane;
- a distance scanner that is used to obtain noisy distances and angular values from the robot to other objects in the environment. Its range is 1.5 meters.
- a range and bearing communication system [9], with which a robot can communicate with other robots that are in line of sight. Its range is 4 meters.

- a gripper, that is used to physically connect to the transported object;
- two wheels independently controlled to set the speed of the robot.

Currently, the marXbots robots are able to work in teams where they coordinate based on simple interactions on group tasks. For example, a group of marXbots robots may collectively move a relatively heavy object from point **A** to point **B** by using their grippers.

##### 4.2 Self-adaptive Behavior for marXbot

To illustrate autonomic behavior based on this approach, let us suppose that we have a robot that carries items from point **A** to point **B** by using two possible routes - route one and route two (see Figure 3).

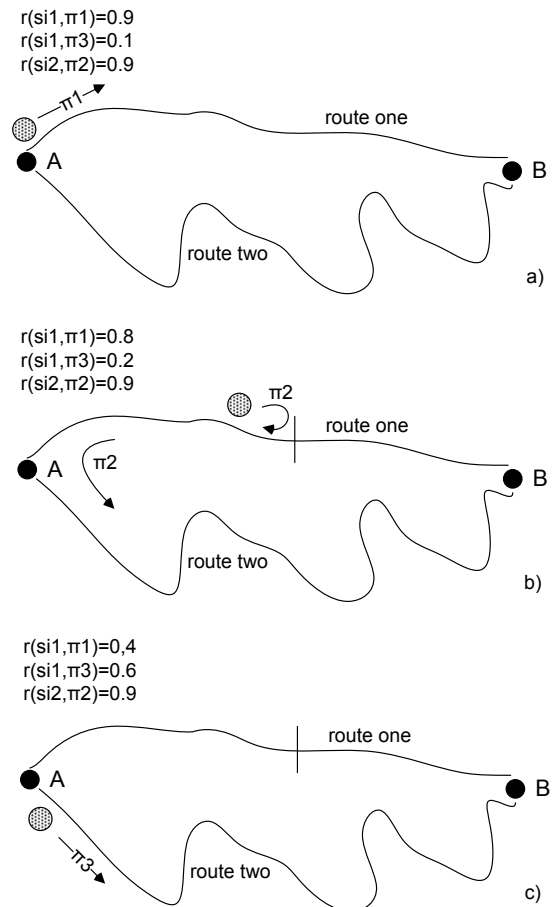


Figure 3. A marXbot Self-adaptation Case Study

A situation  $si1$ : “robot is in point A and loaded with items” will trigger a policy  $\pi_1$ : “go to point B via route one” if the relation  $r(si1, \pi_1)$  has the higher probabilistic belief rate (let’s assume that such a rate has been initially given to this relation because route one is shorter – see Figure 3.a). Any time when the robot gets into situation  $si1$  it will continue applying the  $\pi_1$  policy until it gets into a situation  $si2$ : “route one is blocked” while applying that policy. The  $si2$  situation will trigger a policy  $\pi_2$ : “go back to  $si1$  and then apply policy  $\pi_3$ ” (see Figure 3.b). Policy  $\pi_3$  is

defined as  $\pi_3$ : “go to point *B* via *route two*”. The unsuccessful application of policy  $\pi_1$  will decrease the probabilistic belief rate of relation  $r(si1, \pi_1)$  and the eventual successful application of policy  $\pi_3$  will increase the probabilistic belief rate of relation  $r(si1, \pi_3)$  (see Figure 3.b). Thus, if *route one* continues to be blocked in the future, the relation  $r(si1, \pi_3)$  will get to have a higher probabilistic belief rate than the relation  $r(si1, \pi_1)$  and the robot will change its behavior by choosing *route two* as a primary route (see Figure 3.c). Similarly, this situation can change in response to external stimuli, e.g., route two got blocked or a “*route one* is obstacle-free” message is received by the robot.

## 5. CONCLUSION

This paper has presented an approach to KR&R (knowledge representation and reasoning) for self-adaptive behavior in autonomic systems. The problem is tackled by a framework called KnowLang implying a multi-tier specification model that allows for integration of ontologies together with rules and Bayesian networks. The self-adaptive behavior is achieved via learning based on past experience. The experience is associated with the success of the actions generated in the physical environment by an autonomic system. Maintaining an execution history of these actions helps that system compute the success probability for those actions and eventually learn (infer new knowledge) not to execute actions that traditionally have low success rate. The goal is efficient and comprehensive knowledge structuring and awareness based on logical and statistical reasoning by handling uncertain knowledge where additive probabilities are used to represent degrees of belief.

KnowLang is a formal language for knowledge representation in ASCENS (Autonomic Service-Component ENsemble) systems. As a proof of concept, we applied our approach to build a formal KR&R model for self-adaptive behavior in swarm robotics systems, one of the ASCENS case studies. This formal model has been presented in the paper.

Note that KnowLang is still under development as part of the ASCENS international European project [3]. Our plans for future work are mainly concerned with further and complete development of KnowLang including a toolset for formal validation. Once fully implemented, KnowLang will be used to specify knowledge representation and autonomic behavior in ASCENS case studies.

## 6. ACKNOWLEDGMENTS

This work was supported by the European Union FP7 Integrated Project Autonomic Service-Component Ensembles (ASCENS) and by Science Foundation Ireland grant 03/CE2/I303\_1 to Lero—the Irish Software Engineering Research Centre at University of Limerick, Ireland.

## 7. REFERENCES

- [1] Robinson, P.N. and Bauer, S. 2011. *Introduction to Bio-Ontologies*. CRC Press.
- [2] Neapolitan, R. 2003. *Learning Bayesian Networks*. Prentice Hall.
- [3] ASCENS – Autonomic Service-Component Ensembles. 2010. <http://www.ascens-ist.eu/>.
- [4] European Commission – CORDIS. *Seventh Framework Program (FP7)*. [http://cordis.europa.eu/fp7/home\\_en.html](http://cordis.europa.eu/fp7/home_en.html).
- [5] Vassev, E. and Hinchey, M. 2012. Knowledge Representation for Cognitive Robotic Systems. In *Proceedings of the 15th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing Workshops (ISCORCW 2012)*. IEEE Computer Society. pp. 156-163.
- [6] Vassev, E. and Hinchey, M. 2011. Knowledge Representation and Awareness in Autonomic Service-Component Ensembles – State of the Art. In *Proceedings of the 14th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing Workshops*. IEEE Computer Society. pp. 110–119.
- [7] Vassev, E. and Hinchey, M. 2010. The Challenge of Developing Autonomic Systems. *IEEE Computer*. 43, 12 (December 2010), pp. 93–96.
- [8] Bonani, M., Longchamp, V., Magnenat, S., Retornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H. and Mondada, F. 2010. The MarXbot, a Miniature Mobile Robot Opening new Perspectives for the Collective-robotic Research. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*.
- [9] Roberts, J., Stirling, T., Zufferey, J.-C., and Floreano, D. 2009. 2.5D infrared range and bearing system for collective robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*. pp. 3659–3664.