What Can Ontologies Do for Robot Design?

Francisco Ramos[1]**, Alberto Olivares-Alarcos[2], Andrés Salomón Vázquez[1], and Raúl Fernández[1]

[1] E.T.S. Ingenieros Industriales, UCLM, Ciudad Real, Spain,
[2] **Facultat d'Informàtica de Barcelona, UPC, Barcelona, Spain**

Abstract. In this paper we address the problem of automatic design of the abstract structure of a robot. The design is driven by the desired capabilities that the robot should be able to perform. To this aim, an extension for the IEEE Standard Ontology for Robotics and Automation has been developed. We present an intelligent system which infers abstract robot morphologies from this ontology by relating robot actions to necessary structural parts. Then, these abstract structures can be materialized into physical robots that are able to perform requested capabilities. We show this implementation using a modular robotics platform as a demonstrator.

Keywords: ontologies, robotics, modular robots, robot design

1 Introduction

Designing and controlling a robot to properly work on a given task is the fundamental problem of robotics.

The traditional way to address this problem consists on a two-step process. First, the hardware (i.e. the mechanical structure along with actuating and sensing systems) is designed according to the objective. Then, a specific controller is programmed to achieve the task in the most efficient way. As a result of decades of development, we can find different robot configurations (e.g. manipulators, wheeled robots, humanoids, etc.) considered good enough to perform certain tasks (e.g. manufacturing, exploration, etc.).

In the last two decades a different approach has been explored thoroughly: automatic hardware/software design using intelligent systems. Evolutionary Robotics (ER) is the best example of this approach, where evolutionary algorithms are used to obtain complex mechanisms without having explicit models. A complete review of this line of research in ERs can be found in [6, 16].

Modular robotic platforms are often used in many ER systems as a tool to find and implement the automatic designs [12]. In fact, modular robots are, by themselves, an important area of research in robotics since the late 80s [18, 8].

** Corresponding author: Francisco.Ramos@uclm.es

This work is partly supported by the Centre for the Development of Industrial Technologies (CDTI) and the company Mundo Reader S.L. under the project *BOTBLOQ: Integral ecosystem for the design, manufacturing and programming of DIY robots.*

Theoretically, these architectures might be able to implement any design. This is the reason why they are used in many other areas such as manipulation [2] or educational robotics [7].

In this work we explore the automatic design of the conceptual structure of robots assisted with ontologies for robotics. Our main goal is to demonstrate that the knowledge embedded in an ontology can be used to help a robot design system with small computational effort. As a proof of concept, we present an intelligent system able to select an appropriate *base configuration* depending on the user-requested capabilities that the robot should be able to perform. This configuration lists the physical elements that the robot must contain, but (to date) it does not provides information or constraints on the physical structure. This information must then be used by an intelligent system (or user) in order to generate the hardware and software design for building the robot. For implementation we use ParMoR (Parametric Modular Robots), a low-cost, modular robotics platform intended for use in education. Fig. 1 shows two examples of real robots generated from these base configurations.

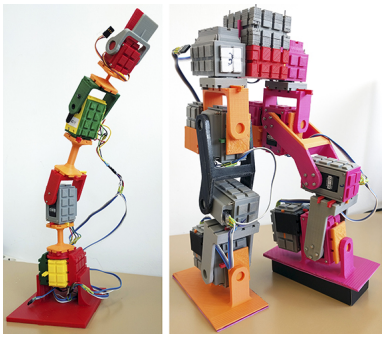

Fig. 1. a) 12DOF legged robot, b) 3DOF manipulator

2 Why ontologies?

In any human process, from simple daily tasks to complex projects, the knowledge, both declarative (*know what*) and procedural (*know how*), is crucial. Thus, when designing machines which are able to face human tasks it is necessary to represent the knowledge behind those tasks in a way they can understand (e.g. ontologies).

Ontologies, in a computational sense, are formal and explicit specifications of conceptualizations [9] and provide enough concepts and relations to articulate models of specific situations in a given domain. However, they do not simply represent but also generate knowledge: if an ontology is correctly designed, we do not need to specify all facts explicitly, but we can also infer implicit knowledge

making it explicit. For instance, if we define the class *HumanoidRobot* as a robot which includes two arms and two legs, when we instantiate that class we do not need to specify that our instance has two arms and two legs. Consequently, for defining real models, we can use an ontology that specifies the conceptualization shared by the experts in the field.

A good example of how ontologies could help to a design process is explained in [10], where the authors tried to bridge what they called 'gaps of knowledge' in the process of designing electronic hardware. Usually, hardware design is divided into different stages but it is not always clear how the relevant parts of each stage are related to the rest of them. For instance, once we have defined user requirements we need to design a system which has suitable capabilities to fit those requirements. At this point, we need to bridge some 'gaps of knowledge' about which kind of capabilities are related to user requirements and which physical parts of our system needs to have in order to perform those capabilities. Those gaps are traditionally handled by human engineers which are likely to produce errors and mismatches in the design process. [10] handle those gaps using the knowledge defined in the ontology.

Another example of success can be found in the medical domain, where the use of ontologies is widely extended. In fact, there even exists a collective of ontology developers that are committed to collaboration and adherence to shared principles: Open Biomedical Ontologies (OBO) Foundry [17]. The mission of the OBO Foundry is to develop a family of interoperable ontologies that are both logically well-formed and scientifically accurate. Some applications are: measurement of semantic similaritiy between medical entities[14, 15, 4] and advisory (expert) systems [11, 5].

2.1 Ontologies for Robotics

The use of ontologies is not new in the domain of Robotics, as there are several applications in which knowledge representation becomes essential. For example, KnowRob [19] is a knowledge processing system that combines knowledge representation and reasoning methods with techniques for inferring new knowledge and for grounding it in a physical system. KnowRob is able to infer a list of actions to carry out a task (what is called a recipe) and to determine if a certain robot can perform the task according to its abilities (equipped devices). This feature was put to use in the RoboEarth project [21, 20], where a recipe repository was available for any robot connected to the internet, and KnowRob determined the range of tasks that the robot could accomplish depending on its hardware/software. Implicitly, we can see here the relationship between a process (actions a robot has to perform) and the tools (parts of the robot) that the robot will use in order to achieve the goal. This relationship will also be adopted in our methodology: we are interested in inferring which kind of structural parts of a robot are required to perform a specific task or action in order to design a robot which contains them.

An effort for standardization has been recently attempted with the release of the IEEE Standard Ontologies for Robotics and Automation (ORA) [3] in

2015. It self-defines as *"a core ontology that allows for the representation of, reasoning about, and communication of knowledge in the R&A domain"*. ORA is constructed upon the Suggested Upper Merged Ontology (SUMO) [13] and contains the Core Ontology for Robotics and Automation (CORA) which includes the fundamental concepts in the R&A domain as well as their definitions, attributes, constraints and relationships. These are required to construct more specific concepts belonging to other ontologies. Actually, ORA also includes the following sub-ontologies:

- CORAX defines concepts too general to be in the CORA ontology and necessary for modeling but not covered by SUMO.
- RPARTS provides an extensible set of the most general and specific types of robot parts.
- POS implements concepts regarding robot poses (both position and orientation concepts).

Nowadays, several working groups of the IEEE are developing extensions of ORA for different domains, such as Industrial Robot Ontology or the Autonomous Robotics (AuR) Ontology, what highlights a consensus in the need and usefulness of a comprehensive, standardized ontology for robotics. However, ORA in its actual state does not match the needs of this work. While it states several concepts that are essential for our design purposes, such as *RobotPart* in CORA or *RobotMotion* in CORAX, it also lacks of specific robot motions and robotic parts that can be interrelated to obtain structural dependencies. Therefore, an extension for the ORA ontologies is required for the automatic design process.

3 Automatic Design of Robots Ontology

The Automatic Design of Robots Ontology (ADROn) defines additional concepts and relations that are to be used for the automatic conceptual design/selection of robots. These concepts are defined in the following subsections.

3.1 Structural Robot Parts

The main physical elements defined in ADROn are *StructuralRobotPart* and *Module*, which are subclasses of *Device* and *Artifact* in SUMO, respectively, as depicted in Fig. 2.

On the one hand, an instance of *StructuralRobotPart* consists of a set of *Modules* and is a *robotPart* of a *Robot* and plays an important role in a specific action of a robot. For example, *RobotLeg* is a *robotPart* of the structure of a robot that is essential to walk or to run (to ambulate). On the other hand, an instance of *Module* will be any artifact (passive or active) which can be related as a part of a *StructuralRobotPart* (e.g. IMU sensors, servomotors, links, etc.).

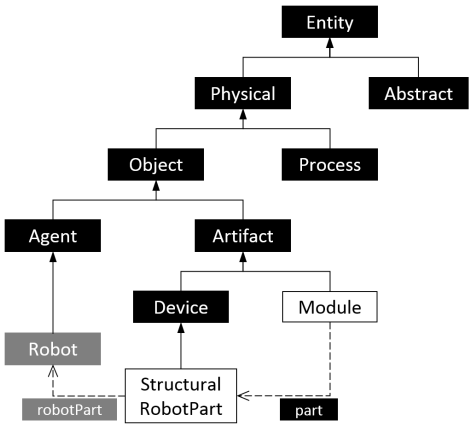

Fig. 2. Taxonomy of the main physical concepts in ADROn (white) and relation with SUMO (black) and CORA (gray).

```
(subclass StructuralRobotPart Device)
(=>
   (instance ?STRUCTURE StructuralRobotPart)
   (exists (?ROBOT)
      (and
         (instance ?ROBOT Robot)
         (robotPart ?STRUCTURE ?ROBOT))))

(subclass Module Artifact)
(=>
   (instance ?MODULE Module)
   (exists (?ROBOT ?STRUCTURE)
      (and
         (instance ?ROBOT Robot)
         (instance ?STRUCTURE StructuralRobotPart)
         (robotPart ?STRUCTURE ?ROBOT)
         (part ?MODULE ?STRUCTURE))))
```

Examples of subclasses of *StructuralRobotPart* are *EndEffector*, *RobotLimb* or *RobotTrunk*.

3.2 Robot Actions

ADROn also defines concepts regarding the actions that a robot can perform under the class *RobotAction*, which is a subclass of *Process* of SUMO and a superclass of *RobotMotion* of CORAX, and refers to a process in which the agent is a *Robot*. Examples of *RobotAction* are *RobotAmbulating* or *RobotLineTracking*.

3.3 Robot Types

ADROn also defines a number of robot types such as *HumanoidRobot* which are subclasses of *Robot* according to the taxonomy outlined in Fig. 4.

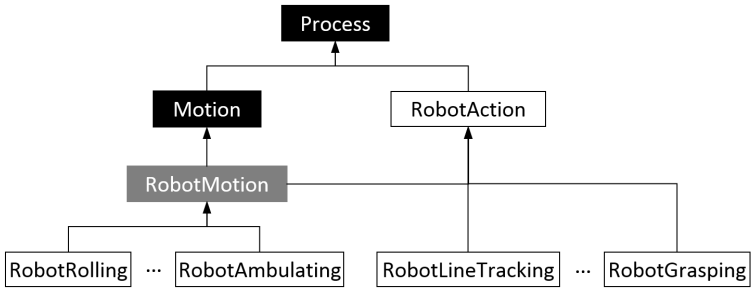

Fig. 3. *RobotAction* declaration and relation with SUMO and CORA axioms.

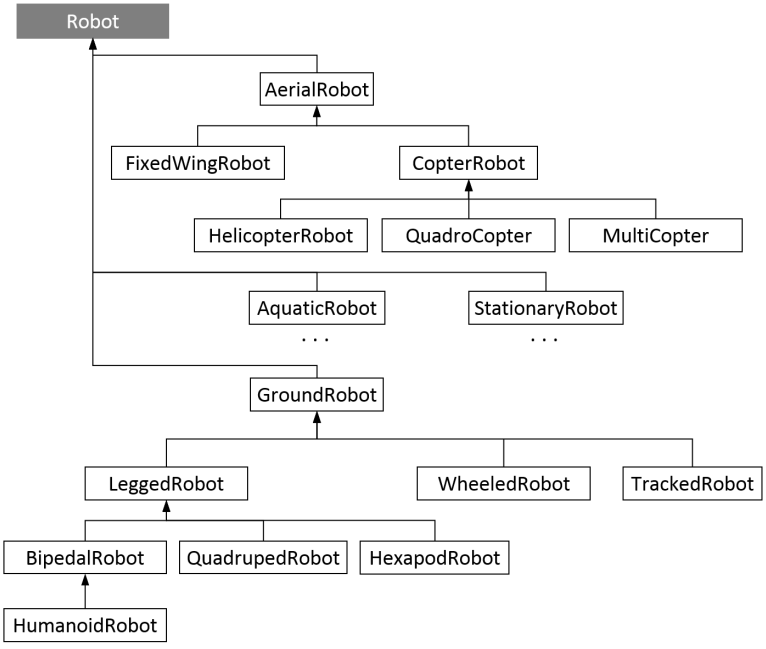

Fig. 4. Partial robot classification depending on the environment/locomotion.

Each of these robots consists of one or more *StructuralRobotParts* depending on their definition. For example, the following code defines a *HumanoidRobot* subclass consisting of two *RobotArms* and a *RobotTrunk* which are, in turn, subclasses of *StructuralRobotParts*. In addition, as long as *HumanoidRobot* is a subclass of *BipedalRobot*, it also includes two instances of *RobotLeg*.

```
(subclass GroundRobot Robot)
(subclass LeggedRobot GroundRobot)
(subclass BipedalRobot LeggedRobot)
(=>
   (instance ?rob BipedalRobot)
   (exists (?leg1 ?leg2 ?trunk)
      (and
         (instance ?leg1 RobotLeg)
         (instance ?leg2 RobotLeg)
         (instance ?trunk RobotTrunk)
         (part ?leg1 ?rob)
         (part ?leg2 ?rob)
         (connectedTo ?leg1 ?trunk)
         (connectedTo ?leg2 ?trunk)
         (part ?trunk ?rob)
         (not (equal (?leg1 ?leg2))))))

(subclass HumanoidRobot BipedalRobot)
(=>
   (instance ?rob HumanoidRobot)
   (exists (?arm1 ?arm2 ?trunk)
      (and
         (instance ?arm1 RobotArm)
         (instance ?arm2 RobotArm)
         (instance ?trunk RobotTrunk)
         (part ?arm1 ?rob)
         (part ?arm2 ?rob)
         (connectedTo ?arm1 ?trunk)
         (connectedTo ?arm2 ?trunk)
         (part ?trunk ?rob)
         (not (equal (?arm1 ?arm2))))))
```

3.4 Structural Requirements

Finally, the dependency of a specific *RobotAction* to a specific *StructuralRobotPart* is determined by an instance of a *BinaryPredicate* named *StructuralRequirement*, which is defined as follows:

```
(instance StructuralRequirement BinaryPredicate)
(instance StructuralRequirement InheritableRelation)
(domain StructuralRequirement 1 RobotAction)
(domain StructuralRequirement 2 StructuralRobotPart)
```

Hence, a *StructuralRequirement* determines, for example, that if a *Robot* is going to grasp an object, it needs a specific *StructuralRobotPart* to do it (e.g. a robot gripper).

```
(subclass RobotGrasping RobotAction)
(subclass RobotGrasping Grabbing)
(=>
   (instance ?GRASP RobotGrasping)
   (exists (?GRIPPER)
      (and
         (instance ?GRIPPER RoboticGripper)
         (StructuralRequirement ?GRASP ?GRIPPER))))
```

4 Robot Instance Generation

According to ADROn, a robot consists of one or more *StructuralRobotParts* and each of them has one or a set of *Modules*. Fig. 5 shows: an instance of a *HumanoidRobot* (subclass of *Robot*); an instance of one of its *RobotLegs* (subclass of *StructuralRobotPart*); and some instances of the *Modules* (active and passive) that constitute the *RobotLeg* in the ParMoR architecture.

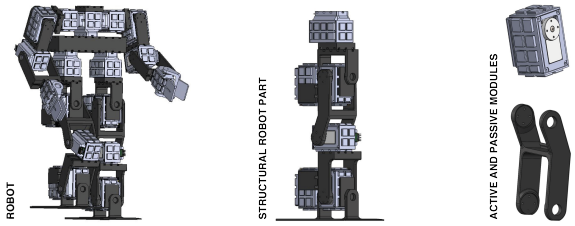

Fig. 5. Instances of the main axioms in ADROn related to the main concept in CORA (*Robot*).

ADROn includes the definition of every module of ParMoR (e.g. *IRProximitySensor*, *Servomotor*, etc.) along with every action that a robot can perform and the relationships between *RobotActions* and *StructuralRobotParts*.

The conceptual generation of a robot is a three-step process. First, the robot instance generator receives a set of *RobotActions* that the robot is required to perform. Then the inference engine uses semantic queries to determine the *StructuralRequirements* implied by the set of actions. Subsequently, the instance generator matches these requirements with the hardware available in the base configurations defined in ADROn. If several matches are found, the generator asks the user some questions inferred from the ontology to disambiguate the solution. Finally, the system creates the conceptual (instance) design of a robot able to perform those actions and passes it to the structure generator. This process is schematized in Procedure 1.

As an example, we present now a very straightforward example of generation of a robot with the ability of *RobotWalking*.

1. First, the generator user demands a robot with a walking capability.
2. A querying process in ADROn determines *RobotLeg* as a *StructuralRequirement*.
3. A search through the base configurations obtains all the matches: *HumanoidRobot*, *QuadrupedRobot* and *HexapodRobot*.
4. The generator asks the user "'Does the walking robot need to grasp?"'.
5. The answer is positive, so the generator determines *RobotGripper* as a new *StructuralRequirement*.
6. It searches again through previous matches and determines that the appropriate robot is a *HumanoidRobot*.

Procedure 1 Robot Instance Generation

Require: sRA ← set of *RobotActions*
  for each *RobotAction* in sRA **do**
    determine *StructuralRequirement*
  end for each
  solutions ← match all *StructuralRequirements*
  while solutions.length > 1 **do**
    generate disambiguation question
    update *StructuralRequirements*
    solutions ← match all *StructuralRequirements*
  end while
  return instance of solution

7. The generator provides a base configuration of a *HumanoidRobot*. The conceptual design is over and the parameterization process begins.

5 Proof of Concept

In the following we present an example of the complete process, summarized in Fig. 6, for the automatic design of a robot using our approach.

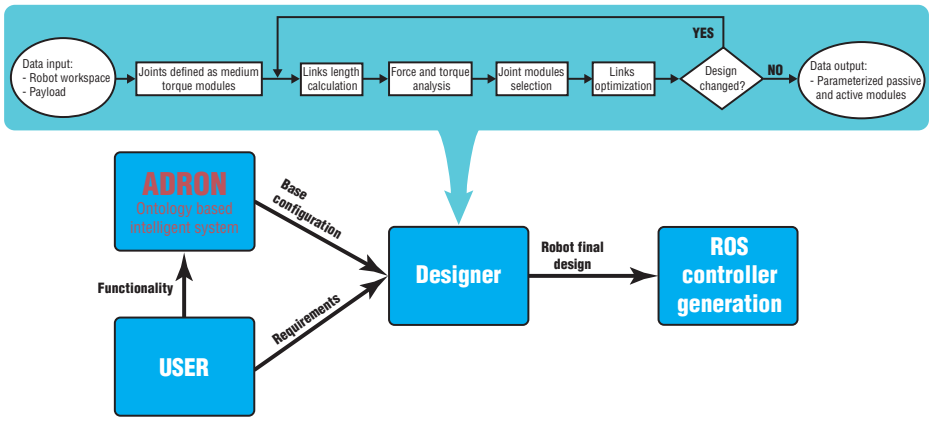

Fig. 6. Process for the automatic generation of robots.

Although in this example we present only the design of a manipulator robot, our system allows users the design of other robots like rovers, snakes, hexapods and humanoids. These robots are finally built using ParMoR, which is a modular robot architecture based on 3D printable modules, that can be active (i.e. contain electronics as shown in Fig. 7a) or passive (i.e. links). As seen in Fig. 7b, modules can be interconnected using dovetail pins.

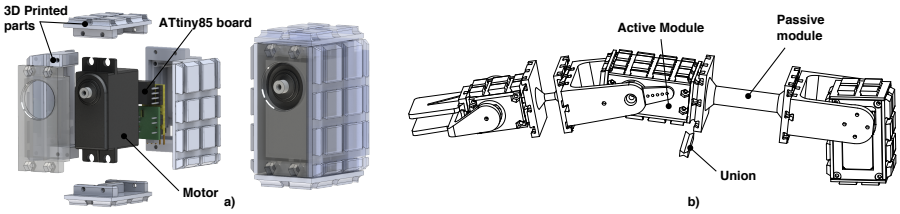

Fig. 7. a) Example of an active module b) Modular Robot Architecture

1. The user, using the GUI of Fig. 8, describes the function to be performed by the robot.

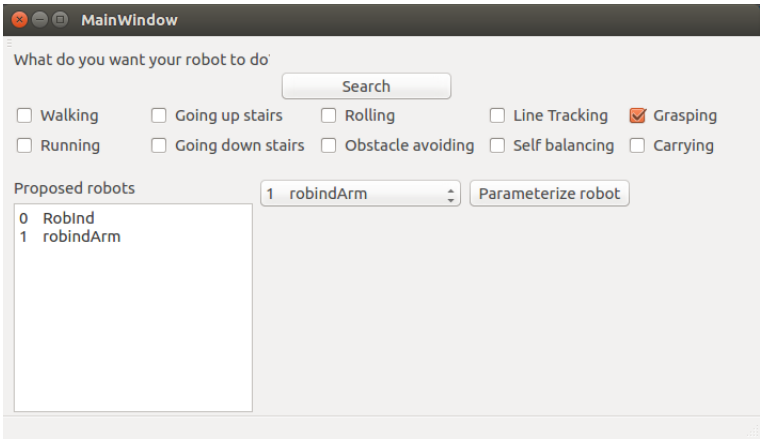

Fig. 8. GUI for the automatic design of robots using ADRON.

2. An intelligent system uses ADROn to infer a robot base configuration as explained in Section 4.
3. The base configuration together with the user requirements (e.g. robot speed, payload, work space) are passed to the automatic designer (see Fig.6) which selects the concrete active modules and the passive modules according to kinematic and dynamic considerations. Fig. 9 shows two different instances of the same base configuration due to a different workspace requirement from the user.
4. The *STL* files for active and passive modules are obtained so they can be 3D printed.
5. Finally, the formal description of the robots is represented in ROS using URDF models [1] and a parameterized controller is provided to simplify the control stage.

Fig. 9. Left manipulator needs two high-torque and one middle-torque actuators, while right manipulator (smaller) needs one high-torque and two medium-torque actuators.

6 Conclusions

Well-designed knowledge representation has given good results in helping the processes of designing electronic hardware or diagnosing diseases. In this communication we have explored the use of ontologies as a supporting tool for robot design.

An extension of the ORA standard (ADROn) has been presented. It provides additional definitions for: structural parts of the robots, actions achievable by a robot, detailed robot types and relations between actions and structural parts. These definitions allow the definition of a procedure for determining the most adequate robot type to perform a given set of actions. The robot instance created can be passed to a physical structure generator in order to create a complete design of a robot. We have presented the implementation of a robot using the ParMoR architecture as a proof of concept.

Presented ontology is in constant evolution, increasing the number of robot types, structural parts and robot actions defined in it, and providing new features. The authors are currently working in an instance generator that create robot instances without replicating a base configuration, but adequately connecting the structural requirements obtained from the requested set of actions. This requires adding physical constraints in the ontology for connection of the different structural parts (e.g. legs must be attached to a trunk).

References

1. Robot Operating System. http://www.ros.org/. Accessed: 2016-09-1
2. Schunk modular robotic system. http://mobile.schunk-microsite.com/en/produkte/products/dextrous-lightweight-arm-lwa-4d.html. Accessed: 2016-09-1
3. Ieee standard ontologies for robotics and automation (2015). URL http://ieeexplore.ieee.org/document/7084073/
4. Batet, M., Sánchez, D., Valls, A.: An ontology-based measure to compute semantic similarity in biomedicine. Journal of biomedical informatics **44**(1), 118–125 (2011)

5. Chen, R.C., Huang, Y.H., Bau, C.T., Chen, S.M.: A recommendation system based on domain ontology and swrl for anti-diabetic drugs selection. Expert Systems with Applications **39**(4), 3995–4006 (2012)
6. Doncieux, S., Mouret, J.B., Bredeche, N., Padois, V.: Evolutionary Robotics: Exploring New Horizons, pp. 3–25. Springer Berlin Heidelberg (2011)
7. Golovinsky, A., Yim, M., Zhang, Y., Eldershaw, C., Duff, D.: Polybot and polykinetic system: a modular robotic platform for education. In: Proceedings 2004 IEEE International Conference on Robotics and Automation (2004)
8. Gonzalez-Gomez, J., Zhang, H., Boemo, E., Zhang, J.: Locomotion capabilities of a modular robot with eight pitch-yaw-connecting modules. In: 9th International Conference on Climbing and Walking Robots (2006)
9. Guarino, N., Oberle, D., Staab, S.: What Is an Ontology?, pp. 1–17. Springer Berlin Heidelberg (2009)
10. Hu, H., Liu, D.y., Du, X.y.: Semi-automatic hardware design using ontologies. In: Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th, vol. 2, pp. 792–797. IEEE (2004)
11. Jonquet, C., Musen, M.A., Shah, N.H.: Building a biomedical ontology recommender web service. Journal of biomedical semantics **1**(1), S1 (2010)
12. Li, H., Wei, H., Xiao, J., Wang, T.: Co-evolution framework of swarm self-assembly robots. Neurocomputing **148**, 112–121 (2014)
13. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of the international conference on Formal Ontology in Information Systems, pp. 2–9. ACM (2001)
14. Pedersen, T., Pakhomov, S.V., Patwardhan, S., Chute, C.G.: Measures of semantic similarity and relatedness in the biomedical domain. Journal of biomedical informatics **40**(3), 288–299 (2007)
15. Pesquita, C., Faria, D., Falcao, A.O., Lord, P., Couto, F.M.: Semantic similarity in biomedical ontologies. PLoS computational biology **5**(7), e1000,443 (2009)
16. Silva, F., Duarte, M., Correia, L., Oliveira, S.M., Christensen, A.L.: Open issues in evolutionary robotics. Evolutionary Computation **24**, 205–236 (2016)
17. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., et al.: The obo foundry: coordinated evolution of ontologies to support biomedical data integration. Nature biotechnology **25**(11), 1251–1255 (2007)
18. Sprowitz, A., Moeckel, R., Vespignani, M., Bonardi, S., Ijspeert, A.: Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. Robotics and Autonomous Systems **62**, 1016–1033 (2014)
19. Tenorth, M., Beetz, M.: KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. International Journal of Robotics Research **32**(5), 566 – 590 (2013). URL http://ijr.sagepub.com/content/32/5/566.short
20. Tenorth, M., Perzylo, A., Lafrenz, R., Beetz, M.: Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework. Automation Science and Engineering, IEEE Transactions on **10**(3), 643–651 (2013). DOI 10.1109/TASE.2013.2244883
21. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: Roboearth. Robotics Automation Magazine, IEEE **18**(2), 69–82 (2011). DOI 10.1109/MRA.2011.941632