Skill-oriented designer of conceptual robotic structures*

Francisco Ramos[1], *Member IEEE*, Cristian O. Scrob[2], Andrés S. Vázquez[1], Raúl Fernández[1] and Alberto Olivares-Alarcos[3].

***Abstract*— This communication presents an application for the use of ontologies in the generation of robot structures. The ontology developed for this app relies on the IEEE Standard Ontologies for Robotics and Automation (ORA) and it incorporates a set of concepts, relations and axioms that link robotic skills with the structural parts needed for their realization. The user can select a base configuration and/or a set of desired skills that the robot should be able to perform. Then, the application evaluates the axioms and returns an abstract structure that can carry out the requested skills. The final implementation of the structure can be achieved with any modular robotic platform that could identify each structural part with a physical device.**

I. INTRODUCTION

Traditionally, robot design has been based on performing a certain task most efficiently and accurately. This design usually consisted of some hardware specifically designed for the task at hand by a group of engineers, and a software heavily oriented to support that specific hardware and to provide only the original design functionalities. In fact, these designs were (more often than not) isolated from, or at least incompatible with, other similar devices.

During the last years, several approaches have tried to address the automation of this design process. Actually, this is a topical issue in robotics which has been addressed in a Workshop at recent ICRA 2018. In [1], robot arm design requirements are encoded as desired motion trajectories for the end-effector. The proposed system enables on-demand design of custom robot arms using a library of modular and reconfigurable parts such as actuators and connecting links. The method generates a functional, as-simple-as-possible robot arm that is capable of tracking the desired trajectories. In [2], formalization, minimality and integration are proposed as other questions that must be addressed in automatic robot design. Nonetheless, the most renowned approaches are based on evolutionary techniques [3], [4]. Evolutionary robot design aims to create robots according to a certain set of initial specifications (inputs), which might be skills, but, instead of a group of engineers, it uses evolutionary algorithms to automate the process. This allows to obtain different solutions for the same inputs, some of which could be devised by humans, while others could be highly innovative or even game changers for robot design. However, in [5], authors state that, while evolutionary robotics has aimed to optimize robot control and morphology to produce better and more robust robots, most previous research only addresses optimization of control, and does this only in simulation. In order to solve this problem, they developed a four-legged mammal-inspired robot that features a self-reconfiguring morphology. Thus, they discuss the advantages and drawbacks of being able to efficiently do experiments on a changing morphology in the real world.

On the other hand, there is a growing interest in the robotics community on the topic of knowledge representation using ontologies. The publication of the IEEE-ORA Standard [6] in 2015 has been a starting point for many applications. For example, in [7] the authors experimentally demonstrated the applicability of ORA. Ontologies can also help with the relational representation of the environment, allowing for example the *semantic navigation* of robots [8], or with the cognitive representation in social robotics [9]. The standard is being extended by the Autonomous Robotics (AuR) Ontology Working Group of the IEEE [10].

In [11], design is defined as the process of satisfying rather than optimizing. However, in our approach, both processes are present during design: evolutionary robotics might be the optimization tool of the designer, while ontologies could help to satisfy the semantic requirements of the user, provided in natural language. Ontologies add the semantic information needed for giving a mechanical structure a meaning, relating their structural parts to the set of skills enabled by having them. This process provides abstract able configurations and, with them, evolutionary algorithms could explore the space of solutions given by the set of physical devices that could replace each structural part using, for example, a modular robotic platform. Specifically, the directed graph of a robot structure, which can be represented using an ontology, could be seen as the representation of solutions/individuals in evolutionary computation methods.

In [12] the authors presented a robot configuration selector based on desired tasks to be performed by a robot. The selection was performed by matching the desired tasks with the abilities of each base configuration by means of an ontology taxonomy. This work was extended in [13] as a part of an end-to-end educational robots designer operable by non-expert users that uses a modular robotic platform called ParMoR for constructing the physical models.

*This work was supported by CDTI under expedient IDI-20150289 (BOTBLOQ: Ecosistema integral para el diseño, fabricación y programación de robots DIY).

[1]F. Ramos, R. Fernández and A.S. Vázquez are with the School of Engineering of the University of Castilla-La Mancha, Ciudad Real, Spain {francisco.ramos, raul.fernandez, andress.vazquez}@uclm.es.

[2]C.O. Scrob is with Indra Systems S.A., Madrid, Spain no.cristian.fr@gmail.com.

[3]A. Olivares-Alarcos is with the Institute of Robotics and Industrial Informatics of the Polytechnic University of Catalonia, Barcelona, Spain aolivares@iri.upc.edu.

In this communication we present a new design procedure that creates abstract robot structures without the need for a base configuration. Our starting point is the Automatic Design of Robots Ontology (ADROn) presented in [12] which is written in SUO-KIF language to be compatible with ORA[6]. ADROn includes classes for different robot morphologies, robotic skills and devices. To improve its functionalities, we have defined a set of relations and laws that impose restrictions on the robot morphology (e.g. a bipedal robot has two legs) and the robotic skills, (e.g. a robot needs feet for walking). In this manner, we add semantic information to the structural parts used in the robot structure creation from natural language requirements.

An application for editing, parsing and visualizing ontologies and instances of ontologies written in SUO-KIF language has been developed. The application allows several modes of design: configuration-based, skill-based and free. The output of this application will be the SUO-KIF instance with the definition of the robot structure.

II. ADROn

ADROn is an ontology which relies on ORA [6]. The aim of ADROn was to define more specific concepts which were not covered within the standard. Specifically, concepts that allowed the automatic inference of abstract design/selection of a robot given its expected capabilities.

Its potential has already been tested in previous works [12], [13], where ADROn was used to decide, among several predefined base configurations (e.g. humanoid, manipulator, etc.), which one was most adequate to perform a certain set of actions (e.g. walking or grasping), specified as user requirements. The taxonomy included three essential terms of our framework: *RobotAction*$_A$[1], *StructuralRobotPart*$_A$[2] and *RobotType*$_A$. The system used ADROn to infer the physical parts (*StructuralRobotParts*$_A$) needed to carry out those actions and then matched them with the defined base configurations. The base configuration that contained all the *StructuralRobotParts*$_A$ needed to fulfill the user requirements was selected as the solution.

In this work we move one step further by removing the base configurations, so that our framework is able to build a robot step-by-step by assembling several structural parts, where each of them fulfills some of the user requirements. Note that these requirements are just a set of capabilities (actions the robot is able to perform), not the function/purpose of the robot itself. For example, the function of a robot might be playing football, while the set of capabilities (our system requirements) needed to achieve that function would be translocate, avoid obstacles, sense a ball and propel a ball.

Obviously, this adds more flexibility to our framework while complicating the problem. Thus, an extension of ADROn becomes necessary, since the new proposal requires more inference power. In the next sections, we present new concepts, relations and axioms which have been added to our previous version of ADROn.

A. Relations

StructuralRobotParts$_A$ and/or *Devices*$_S$ might be related in different ways: if there is some relative movement between them we say they are articulated (e.g. a leg with a trunk), but for static elements they only need to be connected (e.g. the power supply with an actuator). These are some relations that we have found of interest to work with in the ontology:

- *robotPart*$_O$ states that a *StructuralRobotPart*$_A$ belongs to a certain *Robot*$_O$ instance. It is an ORA concept but very useful in the verification process.
- *robotAbleTo* relates a *Robot*$_O$ with a *RobotSkill*.
- *robotCanSense* indicates that a *Robot*$_O$ can measure a certain *Physical*$_S$ property. It is a subclass of relation *canSense*, which indicates that a *Device*$_S$ can measure a certain *Physical*$_S$ property, and also a subclass of *robotAbleTo*. With these relations we provide that the robot inherits the sensing ability provided by a sensor.
- *formedBy* states that a *Device*$_S$ is composed of others (e.g. an inertial measurement unit is composed of accelerometers and gyroscopes).
- *connectedTo* indicates that two different *StructuralRobotParts*$_A$ are physically connected.
- *articulatedTo* adds a constraint to *connectedTo*, specifying that there is a joint between the *StructuralRobotParts*$_A$ related.

B. Terms

The taxonomy of the terms provided by the ontology is too extensive to be displayed in this communication. Some of the main subclasses are the following:

- *StructuralRobotParts*$_A$ represent any part that has a semantic meaning in a robot. It is basically divided in *RobotHead*, *RobotTrunk* and *RobotLimb*. This last, in turn, is divided in *RobotLeg* and *RobotArm*. Many other robot parts are still to be added (e.g. hands or wings).
- However, other physical elements, such as *RobotJoint* or *EndEffector*, are directly defined as subclasses of *Device*$_S$ as they do not add a semantic meaning.
- *Sensor* is a subclass of *Device*$_S$ which, in turn, has many subclasses that are associated to *RobotSkills*, such as, *LocationSensor*, *DistanceSensor* or *VisualSensor*.
- Other common devices in robotics are grouped under *Actuator*, *ElectricDevice* and *CommunicationDevice* subclasses of *Device*$_S$.
- *RobotSkills* are defined as a subclass of *Skill*$_S$, where the subject is a *Robot*$_O$. The taxonomy for *RobotSkills* is displayed in Figure 1.

C. Axioms

Many axioms have been included in order to verify the correctness of the instances definition. They provide for different ways of defining different structures. For example, we could directly instantiate the *HumanoidRobot*$_A$ type or the system

[1]Subscripts indicate concepts previously included in ontologies: A for previous version of ADROn, O for ORA Standard and S for SUMO. Terms without subscripts are new to ADROn in this paper.

[2]Physical artifact which is part of a robot and plays a necessary role when a robot performs an action (e.g. a *RobotLeg* is essential for a robot to walk).

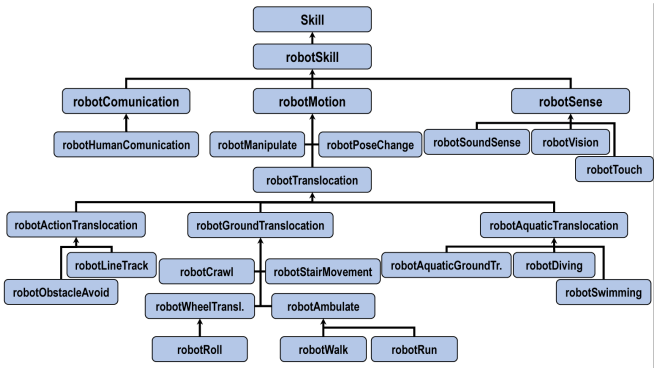

Fig. 1. Taxonomy of the *RobotSkills* included in ADROn.

could automatically extract each one of the components from the definition of $HumanoidRobot_A$. Some sample axioms will be presented, illustrating different situations that our system can cope with.

The following axiom expresses the equivalence of having a *RobotJoint* in between two $StructuralRobotParts_A$ and using the relation *articulatedTo*.

```
(subclass RobotJoint Device)
  (forall (?rJ)
    (=>
      (instance ?rJ RobotJoint)
      (exists (?dev1 ?dev2)
        (and
          (instance ?dev1 Device)
          (instance ?dev2 Device)
          (articulatedTo ?dev1 ?dev2)
          (connectedTo ?dev1 ?rJ)
          (connectedTo ?dev2 ?rJ)
  ) ) ) ) )
```

There are also axioms to consider the structural requirements for a robot to be of a certain type, for example an $HumanoidRobot_A$.

```
(subclass HumanoidRobot BipedalRobot)
  (forall (?rob)
    (=>
      (instance ?rob HumanoidRobot)
      (and
        (exists (?arm1 ?arm2 ?trnk ?head)
          (and
            (instance ?arm1 RobotArm)
            (instance ?arm2 RobotArm)
            (instance ?trunk RobotTrunk)
            (instance ?head RobotHead)
            (articulatedTo ?head ?trunk)
            (articulatedTo ?arm1 ?trunk)
            (articulatedTo ?arm2 ?trunk)
            (robotPart ?trunk ?rob)
  ) ) ) ) ) )
```

Some more elaborated axioms have also been developed. E.g. the line tracking skill axiom states that if there is a physical stimulus that allows to perceive a line, in order to be able to do the following, the robot must have a sensor that can sense that stimulus.

```
  (forall (?rob ?line ?stim ?lflw)
    (=>
      (and
        (instance ?lflw RobotLineTrack)
        (instance ?rob Robot)
        (instance ?line Physical)
        (?lflw ?rob ?line)
        (instance ?stim Physical)
      )
      (and
        (formedBy ?line ?stim)
        (exists (?prt)
          (and
            (instance ?prt Device)
            (canSense ?prt ?stim)
            (robotPart ?prt ?rob)
  ) ) ) ) )
```

III. DESIGN PROCESS

Once we have defined a set of relations, terms and axioms for our ontology, we can establish a design procedure based upon the user requirements. Three different design modes have been considered: configuration-based, skill-based and free mode. This section briefly explains the different design modes and a generic algorithm used for the creation of the robot structure.

A. Configuration-based Mode

In this mode, a robotic configuration is chosen by the user and the system extracts all the characteristics of that configuration and instantiates the needed structural robot parts as defined in the ontology axioms. Any of the robots included in the taxonomy can be fully instantiated with Algorithm 1, where `sA` is the whole set of axioms contained in the ontology. Algorithms 2, 3 and 4 are provided for better understanding of some parts of the complete algorithm.

On top of the robot type, some skills, `sRS`, might also be added as requirements to include other robot parts to our instance (see following subsection for an example).

Algorithm 1 Robot structure design main algorithm

Require: sRS ← set of *RobotSkill* **and** RM ← *RobotMorphology* **and** sA ← set of *axioms*
Create R as an *Instance* of RM
Ask for R to exhibit *sRS*
verify(R, sA)
return instance of solution

Algorithm 2 verify(A,L): Force structure A to verify axioms set L

Require: A ← *Structure* **and** L ← set of *Axioms*
ES ← ∅
for each axiom in L **do**
  if contains(A,axiom.antecedent) **and not** ∃ structure ∈ ES so that contains(structure, axiom.consequent) **then**
    if ∃ structure ∈ ES **and** contains(axiom.consequent, structure) **then**
      Substitute structure with axiom.consequent
    else
      Add axiom.consequent to ES
    end if
  end if
end for each
for each structure in ES **do**
  add(A,structure)
end for each

Algorithm 3 contains(A,B): Verify that logical structure (A) contains (B)

Require: A,B ← *Structure*
  sPA ← set of *parts* ∈ A
  sPB ← set of *parts* ∈ B
  for each pb ∈ sPB **do**

    if ∃ pa ∈ sPA **and** (pa ≡ pb **or** pa is a descendant of pb) **then**
      Discard pa
    else
      return A lacks pb, so it does not contain B
    end if
  end for each
  sRA ← set of *relation* ∈ A
  sRB ← set of *relation* ∈ B
  for each rb ∈ sRB **do**

    if ∃ ra ∈ sRA **and** ra ≡ rb **then**
      Discard ra
    else
      return A lacks rb, so it does not contain B
    end if
  end for each
  return A contains B

Algorithm 4 add(A,B): Add logical structure (B) to (A)

Require: A,B ← *Structure*
  sPA ← set of *Part* ∈ A
  sPB ← set of *Part* ∈ B
  if not (contains(A,B)) **then**

    for each pb ∈ sPB **do**

      if ∃ pa ∈ sPA **and** (pa ≡ pb **or** pa is descendant of pb) **then**
        Discard pa
      else

        if ∃ pa ∈ sPA **and** pa is parent of pb **then**
          Substitute pa with pb
        else
          Add pb to spA
        end if
      end if
    end for each
    sRA ← set of *relation* ∈ A
    sRB ← set of *relation* ∈ B
    for each rB ∈ sRB **do**

      if ∃ ra ∈ sRA **and** ra ≡ rb **then**
        Discard ra
      else
        Create rb ∈ A
      end if
    end for each
  end if
  return A

The characteristics of each $RobotType_A$ are inherited by their descendants. If we considered the taxonomy branch regarding legged robots (see Figure 2):

- A $Robot_O$ has a *RobotTrunk* (as $Robot_O$ is a concept included in ORA, its definition has not been changed and *RobotTrunk* is included in each of the robot subclasses instead).
- A *GroundRobot* is a $Robot_O$ with an undetermined locomotion mechanism to move on the ground.
- A *LeggedRobot* is a *GroundRobot* which uses legs as the locomotion mechanism. It has, at least, one leg.
- A *BipedRobot* is a *LeggedRobot* that has exactly two legs articulated to the *RobotTrunk*.
- A *HumanoidRobot* is a *BipedRobot* that has two arms and one head (resembling a human form), also articulated to the *RobotTrunk*.

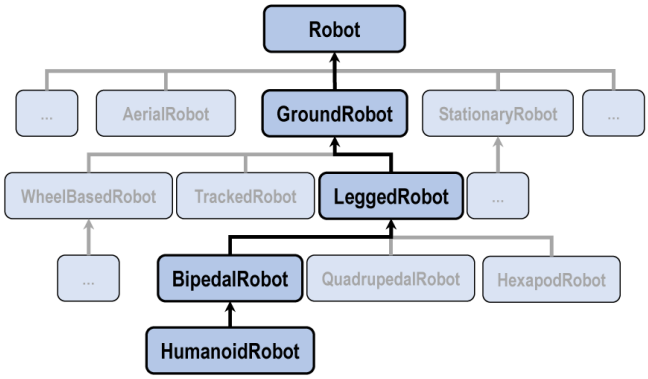

Fig. 2. Taxonomy path from *HumanoidRobot* to root class $Robot_O$.

Then, the robot type creation algorithm moves from the root class, $Robot_O$, to the specific robot type requested adding each of the structural parts included in each subclass.

B. Skill-based Mode

In this mode, the user provides a set of skills as requirements and the robot instance is constructed upon them. The robot structure is created as an instance of $Robot_O$ with a *RobotTrunk*. Subsequently, the rest of $StructuralRobotParts_A$ or $robotParts_O$ are added (Algorithm 4) depending on the fulfillment of the axioms relating certain $Devices_S$/$robotParts_O$ with desired *RobotSkills* (Algorithm 2).

C. Free Mode

In this mode, the user can freely concatenate available $StructuralRobotParts_A$, $robotParts_O$ and $Devices_S$ in any way he considers. The algorithm will check for the fulfillment of the ontology axioms whenever a new element is added (Algorithm 2), as most of these axioms written for the ontology can be used either to infer the missing robot parts to perform a certain skill or to guarantee the consistency of the abstract description of the robot structure.

These three modes could be used jointly if needed (see Section IV-D for an example).

IV. APPLICATION AND CASE STUDY

In order to test the validity of the approach, a graphical application written in JAVA has been developed. It has three functionalities: editor, SUO-KIF parser and robot designer.

A. Parser

Writing an ontology is a tedious task. Other languages such as OWL have useful tools for creating and editing ontologies with friendly interfaces that leverage the task. To help debug our ontology files, a SUO-KIF parser that simplifies the debugging process of the ontology has been written. It indicates the closure of the different elements

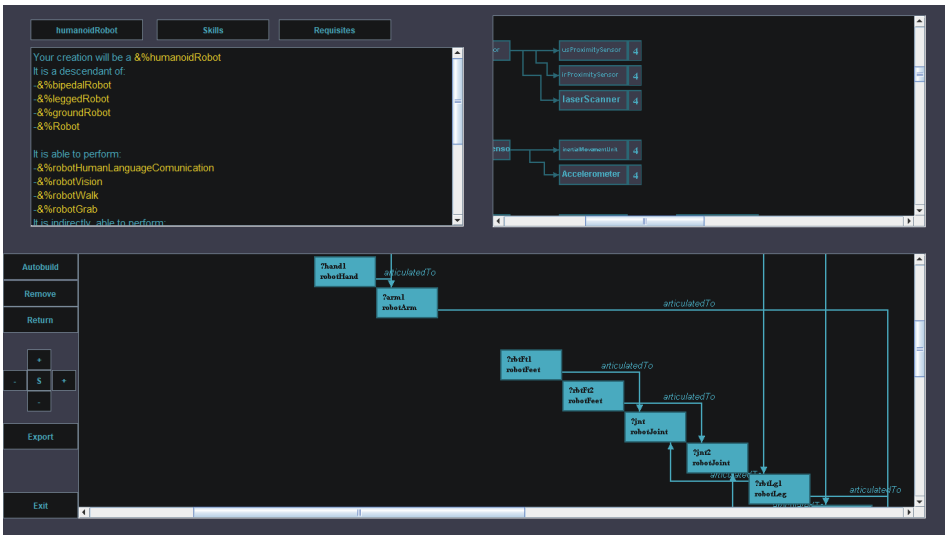

Fig. 3. Screen capture of designer application.

of the language such as instances, subclasses or logical particles. The parser is also responsible for extracting the ontology knowledge into classes and objects that be manipulable within the JAVA code.

B. Editor

As a companion for the parser, a simple text editor with SUO-KIF syntax highlighting has been programmed, to be be used jointly with the parser: the one finds the syntax errors, the other allows file edition simultaneously. In this manner the debugging process is simpler and faster. This is a nice feature as, to our knowledge, no other available editor includes SUO-KIF as a supported language for syntax highlight. Although a style sheet could have been created for some widespread text editors, we wanted to join all functionalities in a single application for the aforementioned practical reasons.

C. Designer

The robot designer allows for the three design modes presented in previous Section. To achieve this, it provides several menus for selecting a robot type (configuration-based), including skills into an existing structure (skill-based) or simply adding any element described in the ontology and attach it to any other robot part (free mode). Figure 3 displays a capture of the designer.

D. Case Study: Humanoid with Skills

In this section we present a case study on the creation of a humanoid structure with a certain set of skills. This will demonstrate the algorithms for creating instances from robot types (*HumanoidRobot*) and for adding skills to a robotic structure.

1) Instantiation of a robot type: When the user selects a certain robot type in the application, Algorithm 1 is executed. For example, if the application is requested to create a *HumanoidRobot* configuration, the algorithm moves from top ($Robot_O$) to bottom (*HumanoidRobot*) through the path of the taxonomy shown in Figure 2. The parts described in the axioms defining each subclass are incorporated, but ignoring already included parts:

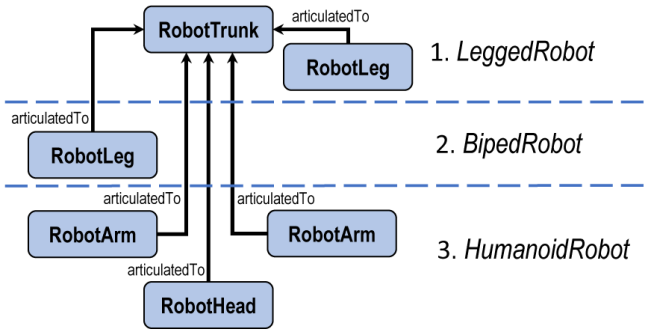

Fig. 4. Instance of a *HumanoidRobot* automatically generated from $RobotType_A$ taxonomy and axioms.

1) As a *LeggedRobot*, the robot must have at least one leg. Therefore, the algorithm creates the *RobotTrunk* and includes a *RobotLeg* that have a relation *articulatedTo* with the *RobotTrunk*.
2) Then, as a *BipedRobot*, the algorithm infers that there should be two *RobotLegs articulatedTo* the *RobotTrunk*. As one of them already exists, only one more is added.
3) Finally, as a *HumanoidRobot* the algorithm inserts two *RobotArms* and a *RobotHead* that have a relation *articulatedTo* with the *RobotTrunk*.

The result is the instance of Figure 4.

2) Addition of skills: After having created the robot configuration, we can add skills to it to provide new capabilities. These skills, in turn, need devices and/or robot parts that must be added to the structure. We have added the following skills to the humanoid robot base:

1) *RobotWalk* requires having feet[3]. These could have an *articulatedTo* relation with the *RobotLegs*, but they can also be *connectedTo* a *RobotJoint* that is, in turn, *connectedTo* the *RobotLeg* (that is exactly the content of the axiom involving *articulatedTo* relation). Also the ability to perform any movement requires that an *ActuatorSystem* be part of the legs and an *ElectricSystem* powering them. Finally, the coordination of the movements of each robot part needs a *ProcessingSystem*.
2) *RobotGrab* demands a mechanism of grasping objects. This could be a *RobotHand articulatedTo* a *RobotArm*. In principle, there is no need for two hands in order to grab something, hence the designer places a single hand in one of the *RobotArms*.
3) *RobotVision* involves a perception system such as a $Camera_S$. This is not a skill by itself, but a requisite for many other skills such as face recognition or object detection which are yet to be added to the ontology.
4) *RobotCommunication*, understood as verbal communication between a human and a robot, needs a $Device_S$ for listening, such as a *Microphone*, and another for talking, such as a *LoudSpeaker*.

The complete robot structure instance is shown in Figure 5.

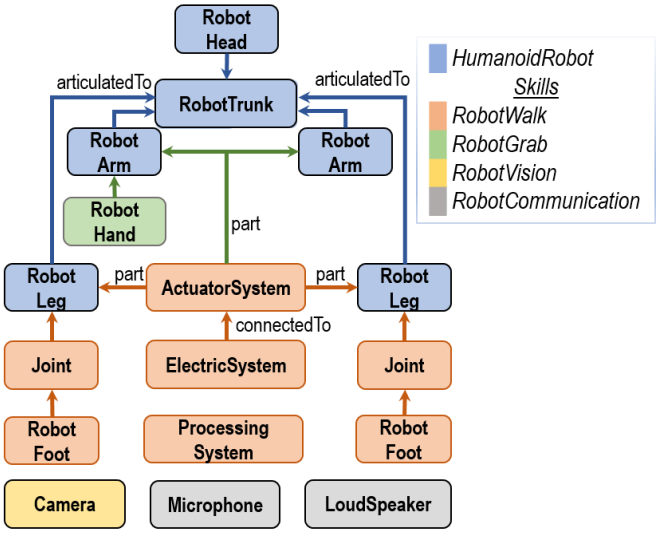

Fig. 5. Instance of a *HumanoidRobot* with a set of skills. The color code indicates the parts added to the robot with each skill (in order).

V. DISCUSSION AND CONCLUSIONS

A procedure for creating conceptual robotic structures have been presented in this paper. It bases upon the knowledge stored in an ontology, which, with an adequate set of axioms on addition to a well structured taxonomy, can be used to generate robots that are able to carry out a certain set of skills demanded by the user.

An application to edit and debug SUO-KIF ontologies has been developed for helping in the process of creation. Also a robot designer GUI has been presented. This designer allows three different modes to creates the conceptual structures of a robot: configuration-based, skills-based and free mode. A beta version of both the application and the ontology can be found in an open GitHub repository[4].

Still, we are aware that there is a considerable gap for these conceptual structures to materialize into physical designs that are effectively able to perform the demanded skills as promised. However, we consider that evolutionary algorithms, taking these structures as their initial population, (jointly with a versatile modular robotic platform such as ParMoR), might be able to find constructable solutions in an automated manner.

[3]Following the Merriam-Webster dictionary definition (*'to move along on foot'*), we assume that the act of walking does need feet. Otherwise, it would be a different type of translocation.

[4]https://github.com/CrisFisher/Ontology-for-Robotics

REFERENCES

[1] R. Desai, M. Safonova, K. Muelling, and S. Coros, "Automatic design of task-specific robotic arms," *arXiv preprint arXiv:1806.07419*, 2018.
[2] A. Q. Nilles, D. A. Shell, and J. M. O'Kane, "Robot design: Formalisms, representations, and the role of the designer," *arXiv preprint arXiv:1806.05157*, 2018.
[3] H. Li, H. Wei, J. Xiao, and T. Wang, "Co-evolution framework of swarm self-assembly robots," *Neurocomputing*, vol. 148, pp. 112–121, 2014.
[4] A. Spielberg, B. Araki, C. R. Sung, R. Tedrake, and D. Rus, "Functional co-optimization of articulated robots," in *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, 2017, pp. 5035–5042. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989587
[5] T. F. Nygaard, C. P. Martin, J. Torresen, and K. Glette, "Exploring mechanically self-reconfiguring robots for autonomous design," *arXiv preprint arXiv:1805.02965*, 2018.
[6] *IEEE Standard Ontologies for Robotics and Automation*, IEEE Std., 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7084073/
[7] V. A. Jorge, V. F. Rey, R. Maffei, S. R. Fiorini, J. L. Carbonera, F. Branchi, J. P. Meireles, G. S. Franco, F. Farina, T. S. da Silva, M. Kolberg, M. Abel, and E. Prestes, "Exploring the IEEE ontology for robotics and automation for heterogeneous agent interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 12 – 20, 2015, special Issue on Knowledge Driven Robotics and Manufacturing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584514000660
[8] J. Crespo, R. Barber, and O. M. Mozos, "Relational model for robotic semantic navigation in indoor environments," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 3, pp. 617–639, Jun 2017. [Online]. Available: https://doi.org/10.1007/s10846-017-0469-x
[9] H. Azevedo, J. P. R. Belo, and R. A. F. Romero, "Cognitive and robotic systems: Speeding up integration and results," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, Nov 2017, pp. 1–6.
[10] S. R. Fiorini, J. Bermejo-Alonso, P. Goncalves, E. P. de Freitas, A. O. Alarcos, J. I. Olszewska, E. Prestes, C. Schlenoff, S. V. Ragavan, S. Redfield, B. Spencer, and H. Li, "A suite of ontologies for robotics and automation," *IEEE Robotics and Automation Magazine*, 2017.
[11] N. Cross, "Designerly ways of knowing: Design discipline versus design science," *Design issues*, vol. 17, no. 3, pp. 49–55, 2001.
[12] F. Ramos, A. Olivares-Alarcos, A. S. Vázquez, and R. Fernández, "What can ontologies do for robot design?" in *ROBOT 2017: Third Iberian Robotics Conference*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Cham: Springer International Publishing, 2018, pp. 465–476.
[13] F. Ramos, A. S. Vázquez, R. Fernández, and A. Olivares-Alarcos, "Ontology based design, control and programming of modular robots," *Integrated Computer-Aided Engineering*, vol. 25, no. 2, pp. 173–192, 2018. [Online]. Available: https://doi.org/10.3233/ICA-180569