Meta-control and Self-Awareness for the UX-1 Autonomous Underwater Robot

Hernandez Corbato, Carlos; Milosevic, Zorana; Olivares, Carmen; Rodriguez, Gonzalo; Rossi, Claudio

DOI
10.1007/978-3-030-35990-4_33

Publication date
2019

Document Version
Final published version

Published in
Robot 2019

Meta-control and Self-Awareness for the UX-1 Autonomous Underwater Robot

Carlos Hernandez Corbato[1(✉)], Zorana Milosevic[2], Carmen Olivares[2], Gonzalo Rodriguez[2], and Claudio Rossi[2]

[1] Delft University of Technology, 2628 CD Delft, The Netherlands
c.h.corbato@tudelft.nl
[2] Centre for Automation and Robotics UPM-CSIC, Madrid, Spain

Abstract. Autonomous underwater robots, such as the UX-1 developed in the UNEXMIN project, need to maintain reliable autonomous operation in hazardous and unknown environments. Because of the lack of any kind of real-time communications with a human operated command and control station, the control architecture needs to be enhanced with mission-level self-diagnosis and self-adaptation properties an additional provided by some kind of supervisory or "metacontrol" component to ensure its reliability. In this paper, we propose an ontological implementation of such component based on Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL). The solution is based on an ontology of the functional architecture of autonomous robots, which allows inferring the effects of the performance of its constituents components in the functions required during the robot mission, and generate the reconfigurations needed to maintain operation reliably. The concept solution has been validated using a hypothetical set of scenarios implemented in an OWL ontology and an OWLAPI-based reasoner, which we aim at validating by integrating the metacontrol reasoning with a realistic simulation of the underwater robot.

1 Introduction

The objective of the European Project UNEXMIN[1] is to develop an underwater vehicle (see Fig. 1) capable of autonomously surveying old mine sites, that are nowadays flooded[2]. The information available regarding the structural layout of the tunnels of such mines is limited, imprecise or even totally lacking. Therefore, prior to any decision, a survey and prospecting of the mine tunnels network should be conducted. Since exploration by human divers is mostly ruled out due to the risks involved, the use of robotic systems appears the only possible solution.

Operating in such environments poses additional requirements, in addition to the "classical" Planning and GNC (Guidance, Navigation and Control) features

[1] H2020, Grant agreement No 690008.

[2] There is a high interest in re-opening some of these sites, since the European Union is largely dependent on raw materials imports.

M. F. Silva et al. (Eds.): ROBOT 2019, AISC 1092, pp. 404–415, 2020.
https://doi.org/10.1007/978-3-030-35990-4_33

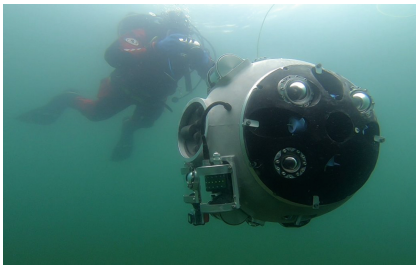

Fig. 1. The UX-1 prototype during a dive in Kaatiala Mine (Finland). Image credits: UNEXMIN consortium, www.unexmin.eu

of autonomous robots. In fact, due to the lack of any kind of real-time communications with a human operated command and control station, the robot, besides of taking autonomous decisions regarding its mission, must be provided with enhanced fault-tolerance capabilities. Here, we decided to go beyond simple "fault tolerance". Our purpose is to augment the UX-1 control architecture with self-adaptation properties to ensure the reliability of its behavior, using the Metacontrol architectural framework by Hernandez et al. [9] and ontological reasoning. The UX-1 perception and motion systems are highly redundant, which allows for multiple (sub-optimal) configurations. Here, we focus on a subset of possibilities for its motion system (see Fig. 2), and demonstrate how, thanks to the self-diagnosis and self-adaptation capabilities provided by the Metacontrol, it can respond to thrusters failures with suitable reconfiguration of both its hardware and software.

We argue that ontological reasoning to drive Metacontrol operation fulfills the needs in the previous context. Ontologies are suitable to capture the system's architecture and capabilities with the appropriate level of abstraction. Ontological reasoning allows to separate the rules for metacontrol operation (i.e. diagnostics and reconfiguration) from the application specific knowledge, and the use of off-the-shelf reasoners, facilitating the validation of the architectural reconfigurations inferred.

This paper presents three novel contributions to the Metacontrol framework: (1) the extension of the Teleological and Ontological Metamodel for Autonomous Systems (TOMASys), to include models of quality attributes, concretely "performance", (2) the use of ontological reasoning for self-diagnostics and reconfiguration, and (3) its proof-of-concept application in the context of the UX-1 autonomous underwater robot.

The paper is organised as follows. Section 2 discusses related research on self-adaptation and ontologies in robotics. Section 3 presents the general Metacontrol framework and the extension of the TOMASys metamodel. Section 4 introduces the specific autonomous underwater robot and its functional architecture, and its ontological modelling with TOMASys. Finally, Sect. 5 discusses the benefits and limitations of solution proposed and addresses next steps in this research and Sect. 6 presents some concluding remarks.

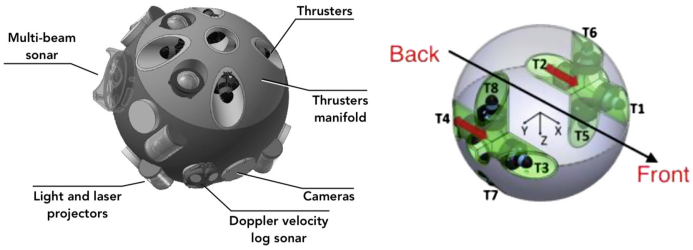

Fig. 2. Left: Main components if the UX-1 robot. Right: thrusters allocation.

2 Related Work

In robotics, ontologies and symbolic reasoning have proven effective for mission management and high-level representation of robot environments and tasks [1], thanks to their ability to represent heterogeneous knowledge. In particular, underwater autonomous robots are a very good testbed for this kind of systems, since their working environment is difficult and hazardous for humans, and hence the need for autonomy and resilience is much more compelling. Zhai *et al.* [18], developed a OWL and SRWL-based ontology to provide users of underwater robots with query services to know the status of the robotic systems and the underwater environment. We use the same technologies, but to implement automatic diagnosis and reconfiguration of an autonomous underwater robot. The Ontology for Autonomous Systems (OASys) [3] is meant to describe and drive the entire life-cycle of autonomous systems, from engineering to operation, base on metamodelling and ontologies.

Control systems of autonomous robots are software systems. The self-adaptation and the use of models at runtime has been extensively studied in software systems in the last decade [2,17]. Simple examples have shown that models of software properties, such as execution time, can be used to optimize the software design of a robot according to associated requirements, such as safety [4,15].

In a previous work [9] we demonstrated how the metamodelling approach in [3] can be used for self-adaptation at the mission level at runtime, going beyond component fault-tolerance. In this work, we advance in that roadmap with the use of ontologies to reason on the robot control architecture and its properties during the mission.

3 Metacontrol Framework

The work presented here extends the framework by Hernandez *et al.* [9] for self-adaptive autonomous systems. The core idea (Fig. 3) is to leverage the engineer knowledge of the system in the form of a runtime model, to drive its runtime self-adaptation capabilities. This knowledge is captured as a model of the functional architecture during system development [7], by using the Integrated Systems

Engineering and PPOOA (ISE&PPOOA) method for Model-Based Systems Engineering (MBSE) [6]. Following the functional approach in ISE&PPOOA allows to raise the level of abstraction in the system's representation and focus on representing the architectural properties that are particularly relevant for the system's capabilities required in the mission at hand. To create an application and domain independent solution, the Metacontrol framework departs from knowledge representation approaches in robotics, centered in application-specific models, by applying a metamodelling approach. The TOMASys metamodel [9] is designed to represent runtime models for metacontrol, based on other metamodels for MBSE (UML, SysML), so that eventually transformations can be defined to automatically generate the runtime model from the functional architecture model.

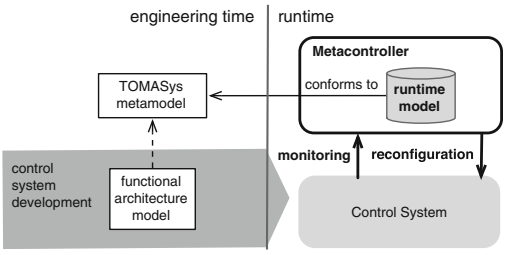

Fig. 3. The Metacontrol approach.

3.1 TOMASys

TOMASys is a metamodel to represent the functional architecture of an autonomous robot, and distinctively also its runtime state. TOMASys functional concepts are based on the theoretical framework for autonomous cognitive systems by Lopez [13], and TOMASys elements related to *structure* are inspired by models for component-based software [14]. Here we will only present briefly the functional elements in TOMASys, displayed in its ontological version in Fig. 4, a complete description of the original TOMASys metamodel can be found in [8].

A differentiating aspect in TOMASys is that it takes the functional approach from systems engineering and functional modelling. TOMASys explicitly captures the relation between the robot's objectives[3] and its control architecture through functional decomposition. At each instant, the system pursues a set of *Objectives*, or specific instances of the functional requirements of the system, i.e. the *Functions*. A *FunctionDesign (FD)* is selected in order to *Solve* each particular *Objective* using concrete system resources, i.e. *Components*. The *FunctionDesign* contains a set of *Roles* that define how certain *Components* in the system are configured to fulfill the Function. The instantiation or grounding of a FunctionDesign by the *Binding* of its defined *Roles* to actual *Components* is

[3] We use italics for the elements in TOMASys ontology.

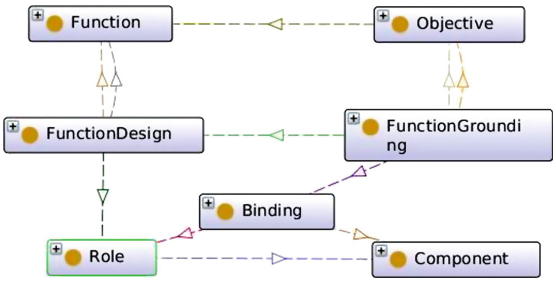

Fig. 4. Main classes in the OWL implementation of TOMASys

represented by a *FunctionGrounding (FG)*. The *FunctionDesign* may also *require* the realisation of other *Objectives*, which would in turn require the instantiation of other *FunctionDesign* that solve them, and so on. This is how TOMASys represents functional decomposition.

Summarily, a TOMASys model represents the control architecture of an autonomous robot through two sets of elements. The first set includes *Objectives*, *FGs*, *Bindings*, *Components* and additional property values and links to capture the information of dynamic state of the functional hierarchy of the system. The second set includes *Functions*, *FDs* and *Roles*, to represent the static knowledge about the system resources and architectural alternatives.

TOMASys originally supported simple fault-based reasoning for diagnosis and component fault-tolerance and functional reconfiguration [9]. This was implemented as fault-propagation in the functional hierarchy model (*FGs* and *Objectives*). In this work we address quality attributes of functions, using performance as an example. For this, we have extended TOMASys with three elements. *Performance* is a property of *Components*, *FunctionGroundings* and *Objectives*, and it is a real number [0.0, 1.0]. *Efficacy* is a property of *FunctionDesigns*, also [0.0, 1.0], that express their ability to solve the objective they address. The instantaneous performance achieved at runtime for each objective in the system depends on the performance of the *FunctionGrounding* solving it and the efficacy of the *FD* it grounds. To compute the performance of the *FG*, different models could be considered, and we have enabled the definition of different performance models for different *FDs* through SWRL rules. However, here we will consider only a default model for all *FDs*, which equals the performance of its grounded *FG* to the average of the performances of the components binded to the roles defined by the *FD*.

3.2 Ontological Reasoning for Metacontrol

One of the core contributions of this work is the implementation of TOMASys in an OWL ontology with SWRL rules. This way, the metacontrol operation for functional diagnosis and reconfiguration is implemented using an OWL reasoner, with the following benefits. First, it facilitates validation and verification by using

standard reasoners for the metacontrol operation, and explicitly separating it from the model semantics. Finally, it opens the door to extend the runtime model with knowledge from other robot ontologies, and enables the use of the multiple ontological tools for metacontrol development.

The Semantic Web Rule Language (SWRL) extends OWL with the capability to specify Horn-like rules (i.e. statements of the form "if-then") to perform inferences over OWL individuals, so new knowledge about those individuals can be generated. An example of the SWRL rules developed to implement TOMASys semantics is shown in Table 1. Concrete, these rules allow to perform functional diagnosis: they identify which objectives are affected by an error in a component. R1 propagates an error detected in a component by the monitoring infrastructure, to the role it plays in a function, by setting the binding's status to `ERROR`. R2 scales that error to the function grounding, and R3 to the objective being realised by the function grounding. Rules 4 and 5 in Table 2 determine that a function design that uses a component that is not available because of being unique and in `ERROR`, is not realisable.

Table 1. Example of the SWRL rules implementing TOMASys semantics for error propagation.

R1	Binding(?b) ^binding_component(?b, ?c) ^c_status(?c, false) ->b_status(?b, false)
R2	Binding(?b) ^hasBindings(?fg, ?b) ^b_status(?b, false) ->fg_status(?fg, false)
R3	Objective(?o) ^fg_status(?fg, false) ^FunctionGrounding(?fg) ^realises(?fg, ?o) ->o_status(?o, false)

Table 2. SWRL rules implementing.

R4	Component(?c) ^typeC(?c, ?cc) ^c_status(?c, false) ^cc_unique(?cc, true) ->cc_availability(?cc, false)
R5	FunctionDesign(?fd) ^ComponentSpecification(?cs) ^cc_availability(?cc, false) ^ Role(?r) ^roleDef(?r, ?cs) ^typeC(?cs, ?cc) ^roles(?fd, ?r) ^ComponentClass(?cc) ->realisability(?fd, false)

We have used a rule-based reasoner that supports SWRL, concretely Pellet [16], and OWLAPI, to implement the metacontrol operation as symbolic inference, in a similar approach to that of [11]. The TOMASys metamodel assumes that the architectural alternatives (FDs) are finite and known, and the statuses of the system components are fully known through monitoring. We use OWLAPI constructs to implement our reasoning with the partial-closed world assumption. The metacontrol reasoning operation consists of functional diagnosis and reconfiguration.

Firstly, the **functional diagnosis** proceeds as follows. The monitoring observations about the system components are converted into assertions of OWL

individuals. Then, the reasoner is executed on the updated ontology, and the status of *FGs* and *Objectives* is inferred using the TOMASys SWRL rules. In this work, in addition to the rules for functional diagnosis in Table 1, SWRL rules are defined for the *FDs* in the application ontology to implement the new TOMASys performance model (see Rule 5 in Table 3). These rules update the performance achieved in the fulfillment of each *Objective*, based on the instantaneous performance of the components involved in their realization[4] and the efficacy of the *FDs* they ground.

Secondly, the metacontrol reasoner infers the best reconfiguration to optimize *Objectives' performance*, based on the available resources, i.e. *Components*, and the design knowledge, i.e. the *Efficacies* of the *FDs* that can be implemented with them. This is done reasoning again with the application-specific rules for performance, in addition to the generic rules for TOMASys semantics.

4 Metacontrol of an Underwater Robot

In this section, we elaborate the application of the Metacontrol framework to implement reasoning for self-adaptation in the control architecture of the UX-1 robot. Concretely, the objective was to design the metacontroller to: (1) perform self-diagnosis of the navigation and motion subsystems, given the status of the thrusters, and (2) determine the best configuration of both subsystems based on the previous diagnosis.

The knowledge obtained during the engineering of both subsystems is that (i) according to the level of performance of the thrusters, different controllers shall be used for optimal performance, and similarly (ii) according to the controller used, one or the other navigation system is suitable. To account for this type of self-adaptation requirements, we have extended TOMASys to model performance as presented at the end of Sect. 3.1.

4.1 Control Architecture of the Underwater Robot

Figure 2 depicts the UX-1 prototype, highlighting both its perception and actuation means. As mentioned earlier, in this work we take into consideration only a subset of all the devices as a proof of concept. Concretely, we take into account only four of the eight thrusters (the ones dedicated to the forward and backward movement), and we consider only the subsystem to control the motions of the underwater robot, including two low-level possible controllers (one PID and one fuzzy) tuned for different thruster configurations. We use the ISE&PPOOA [5,7] to develop the hypothetical model of the control architecture for the motion subsystem and its alternative configurations. This is represented in Fig. 5, where only two of the multiple alternatives are displayed for each function, and the Navigation function is not detailed as it is not considered in the concept proof presented here.

[4] We assume here that this information is provided by the monitoring infrastructure or other specific observers.

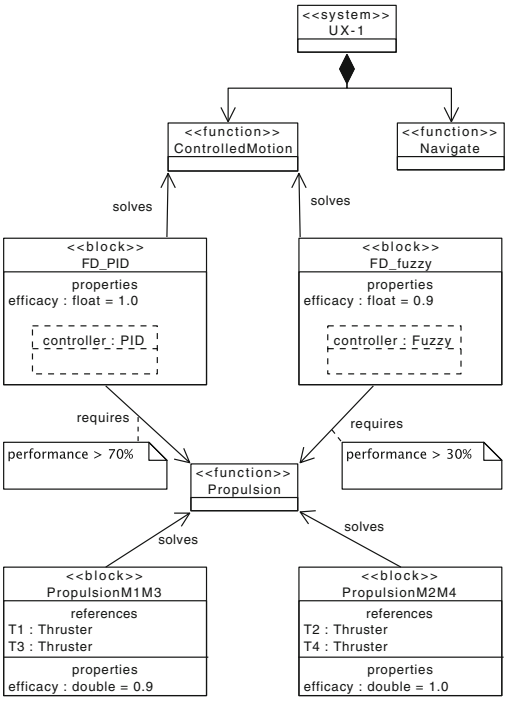

Fig. 5. Architectural alternatives for the UX-1 MotionControl and Propulsion functions, displayed using the SysML graphical notation.

For forward motion, the pair {T2, T4} (front thrusters) is preferred (`efficacy=1`), but any combination of T1/T2–T3/T4 is allowed, depending of the performance of the thrusters. For example, if T1 has bad performance, T2 can be employed. The rationale is that, according to the level of performance of the thrusters, different controllers shall be used for optimal performance. For the shake of our hypothetical scenario, it is assumed The PID controller works well when the performance of the thrusters results in a propulsion thrust above 70%, while the fuzzy controller performs better when the overall thrust is below 30% due to the instantaneous thruster's performance.

4.2 Ontology for the Underwater Robot

For the metacontroller to reason at runtime the optimal configuration of the control architecture of the US-1, the previous engineering knowledge is encoded in the ontology following the TOMASys metamodel.

The UX-1 ontology consist of two modules: the TOMASys ontology, and the application specific ontology for the underwater robot. The later contains the simplified version of the robot's control architecture described in Sect. 4.1.

TOMASys concepts have been implemented through OWL classes, complemented with SWRL rules for additional Metacontrol semantics, in the so called TBox (for terminological knowledge), which is used to represent the domain, i.e. the functional architecture of autonomous systems in this case. The model of the underwater robot's architecture is thus captured in the ABox (for assertions) by creating specific instances of the TOMASys classes for the different objectives, functions and components in the underwater robot's architecture.

In our simplified version of the UX-1, the system has the root *Objective* to `Navigate`, for which any *FunctionDesign* solving it requires the *Objective* `ControlledMotion`. Correspondingly, multiple *FDs* for `ControlledMotion` are available, each one corresponding to each of the different controller options presented in the previous section, and all of them requiring a specific objective for the function `Propulsion`. Finally, the different configurations of thrusters are modeled as different *FDs* for the *Function* `Propulsion`.

We have used the new TOMASys elements *performance* and *efficacy* already discussed in Sect. 3.1, and an associated set of SWRL rules to model the runtime performance of alternative architectures. For example (see Fig. 5), the *FunctionDesign* `FD_PID` solves the function `ControlledMotion` with an efficacy of 1.0 (optimal), and for this it has a role that contains a specification for the PID controller, and requires that an objective of *functionType* `Propulsion` is fulfilled with a performance higher than 0.7. At the lower level, multiple *FDs* are available to address objectives of *functionType* `Propulsion`. For example, `PropulsionM1M3` defines two roles, specifying the use of thruster T1 and T3 for propulsion, with an efficacy of 0.9 (they are not the optimal configuration for propulsion). During runtime operation, the performance achieved for each objective is given by the TOMASys model for performance, which has been implemented in application-specific SWRL rules such as R6 in Table 3. For example, the performance for the propulsion objective, when realized by a *FG* implementing the *FD* `PropulsionM1M3`, is given by multiplying the performance of both thrusters, weighted by the efficacy of the *FD*, which in this case is 0.9. Note that different performance models could have been specified for different *FDs*.

5 Discussion

The previous proof of concept shows the benefits of our metacontrol approach for the self-adaptation of robot control architectures. The ontological implementation of TOMASys allows to implement a general functional diagnosis for robot control architectures, novelly including performance considerations, by adding a model for this quality attribute in TOMASys. The approach followed for the ontological implementation of TOMSys has TOMASys elements implemented in the TBox, as OWL classes and SWRL rules complementing the metacontrol semantics, and the application-specific model represented through individuals in the ABox. This allows for a clear separation of application-specific knowledge and reuse of the TOMASys ontology file and reasoner across applications and domains.

Table 3. One of the SWRL rules that implement the TOMASys performance model for the UX-1.

R6	FunctionGrounding(?fg) ^typeF(?fg, ?fd) ^fd_efficay(?fd, ?eff) hasBindings(?fg, ?bA) ^hasBindings(?fg, ?bB) Binding(?bA) ^binding_role(?bA, role1-fd_move_fw_2m13) ^binding_component(?bA, ?motorA) ^ Binding(?bB) ^binding_role(?bB, role3-fd_move_fw_2m13) ^binding_component(?bB, ?motorB) ^ c_performance(?motorA, ?pA) ^c_performance(?motorB, ?pB) ^ swrlb:add(?aux1 , ?pA, ?pB) ^swrlb:divide(?aux2, ?aux1, 2.0) ^ swrlb:multiply(?eff, ?aux2, .0) -> fg_performance(?fg, ?aux)

The second contribution of this paper is the extension of the TOMASys metamodel to incorporate quality attributes in the model of the functional hierarchy. Previously, TOMASys only accounted for a "confidence" property of the functional design in the architecture, that allowed to propagate component's faults into the functional hierarchy, and diagnose their impact in the system's objectives. This approach was very limited and did not support considering more detailed quality attributes that are usually considered in the engineering of systems, e.g. performance, efficiency/power.

However, the proof-of-concept with the underwater robot have also manifested a difficulty with the current TOMASys metamodel to capture bottom-up design considerations. For example when the performance of some components (controllers) in the robot depends on which other components they are interacting with (thrusters). The solution we have applied is to make the dependency indirect through intermediate objectives (e.g. propulsion). An alternative solution is to create a "flatter" model, in which the interdependent components are captured as multiple roles in a FunctionDesign, having as many *FDs* as alternative configurations for those components.

OWL and SWRL present some limitations for our metacontrol, mainly related to their open-world assumption, that were exposed by the proof of concept. In practice, for TOMASys modelling one of the limitations of SWRL is that it cannot represent rules that require to iterate over individuals. In some cases, e.g. rule 5 in Table 1, we have been able overcome this limitation with the addition of local closed-world assumption by injecting facts with OWLAPI, implementing an application-independent SWRL rule, at the cost of ad-hoc metacontrol reasoning out of the standard reasoner, therefore reducing the original benefits obtained when implementing all the semantics explicitly in the ontology, e.g. verification and validation.

5.1 Future Work

Currently we are testing different ontology engineering approaches for the UX-1 runtime model and associated metacontrol reasoning designs. The representation limitations of TOMASys and OWL/SWRL need to be addressed. Then, we plan to implement the metacontrol operation with a realistic version of the UX-1 control architecture, and test it with a simulation of the underwater robot,

analyzing the influence of the reasoner execution time in real reconfiguration scenarios.

In a next step, we plan to investigate the integration of TOMASys with the ontological standards in robotics CORA (IEEE 1872-2015) [10], which contains concepts to represent the complete triplet mission–system(architecture)–environment. This would allow to coordinate the metacontrol operation with the mission or task-planning. Finally, To address the ontological engineering burden (e.g. modelling of many alternative *FDs*, and their associated SWRL rules) we plan to explore ontology design patterns [12] and metamodelling transformations.

6 Concluding Remarks

In conclusion, we believe that component-level fault-tolerance is insufficient and self-diagnosis and self-adaptation capabilities are needed for autonomous robots, such as the underwater robot UX-1, that need to maintain reliable autonomous operation in hazardous, unknown environments.

The implementation of our Metacontrol framework for self-adaptation using OWL ontologies and SWRL rules for reasoning, extended with specific models of quality attributes of the control architecture, has demonstrated to be effective for this purpose, and the experience gained in this real-world application has raised a series of interesting questions and potential research lines.

Acknowledgements. This work was supported by the UNEXMIN (Grant Agreement No. 690008) and ROSIN (Grant Agreement No. 732287) projects with funding from the European Union's Horizon 2020 research and innovation programme, and has been co-funded by the RoboCity2030-DIH-CM Madrid Robotics Digital Innovation Hub ("Robotica aplicada a la mejora de la calidad de vida de los ciudadanos. fase IV"; S2018/NMT-4331), funded by "Programas de Actividades I+D en la Comunidad de Madrid" and cofunded by Structural Funds of the EU.

References

1. Beetz, M., Beßler, D., Haidu, A., Pomarlan, M., Kaan Bozcuoglu, A., Bartels, G.: KnowRob 2.0 — a 2nd generation knowledge processing framework for cognition-enabled robotic agents, pp. 512–519, May 2018
2. Bencomo, N., Götz, S., Song, H.: Models@run.time: a guided tour of the state of the art and research challenges. Softw. Syst. Model. **18**(5), 3049–3082 (2019)
3. Bermejo-Alonso, J., Hernández, C., Sanz, R.: Model-based engineering of autonomous systems using ontologies and metamodels. In: 2016 IEEE International Symposium on Systems Engineering (ISSE), pp. 1–8, October 2016
4. Brugali, D., Capilla, R., Mirandola, R., Trubiani, C.: Model-based development of QoS-aware reconfigurable autonomous robotic systems. In: 2018 Second IEEE International Conference on Robotic Computing (IRC), pp. 129–136, January 2018
5. Fernandez, J.L., Lopez, J., Gomez, J.P.: Feature article: reengineering the avionics of an unmanned aerial vehicle. IEEE Aerosp. Electron. Syst. Mag. **31**(4), 6–13 (2016)

6. Fernandez-Sánchez, J.L., Hernández, C.: Practical model based systems engineering. Artech House (2019)
7. Hernandez, C., Fernandez-Sanchez, J.L.: Model-based systems engineering to design collaborative robotics applications. In: 2017 IEEE International Systems Engineering Symposium (ISSE), pp. 1–6, October 2017
8. Hernández, C.: Model-based Self-awareness patterns for autonomy. Ph.D. thesis, Universidad Politécnica de Madrid, ETSII, Dpto. Automática, Ing. Electrónica e Informática Industrial, October 2013
9. Hernández, C., Bermejo-Alonso, J., Sanz, R.: A self-adaptation framework based on functional knowledge for augmented autonomy in robots. Integr. Comput. Aided Eng. **25**(2), 157–172 (2018)
10. IEEE Robotics and Automation Society: IEEE Standard Ontologies for Robotics and Automation. Technical report. IEEE Std 1872–2015, February 2015
11. Jajaga, E., Ahmedi, L.: C-SWRL: SWRL for reasoning over stream data. In: 2017 IEEE 11th International Conference on Semantic Computing (ICSC), pp. 395–400, January 2017
12. Krieg-Brückner, B., Mossakowski, T.: Generic ontologies and generic ontology design patterns. In: Workshop on Ontology Design and Patterns (WOP-2017), located at ISWC 2017, Wien, Austria, 21–25 October. CEUR (2017)
13. López, I., Sanz, R., Hernández, C., Hernando, A.: Perception in general autonomous systems. In: Grzech, A. (ed.) Proceedings of the 16th International Conference on Systems Science, vol. 1, pp. 204–210 (2007)
14. OMG: Robotic Technology Component Specification. Technical Report, formal/2008-04-04, Object Management Group, April 2008
15. Ramaswamy, A., Monsuez, B., Tapus, A.: Model-driven self-adaptation of robotics software using probabilistic approach. In: 2015 European Conference on Mobile Robots (ECMR), pp. 1–6, September 2015
16. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. J. Web Semant. **5**(2), 51–53 (2007). Software Engineering and the Semantic Web
17. Weyns, D., et al.: Perpetual assurances for self-adaptive systems. In: de Lemos, R., et al. (eds.) Software Engineering for Self-Adaptive Systems III. Assurances, pp. 31–63. Springer, Cham (2017)
18. Zhai, Z., Martínez Ortega, J.F., Lucas Martínez, N., Castillejo, P.: A rule-based reasoner for underwater robots using OWL and SWRL. Sensors **18**, 3481 (2018)