An Ontology–Based Approach for Autonomous Systems' Description and Engineering

The OASys Framework

Julita Bermejo-Alonso, Ricardo Sanz,
Manuel Rodríguez, and Carlos Hernández

Autonomous Systems Laboratory (ASLab), Universidad Politécnica de Madrid
jbermejo@etsii.upm.es,
{ricardo.sanz,manuel.rodriguezh,carlos.hernandez}@upm.es

Abstract. Ontologies provide a common conceptualisation that can be shared by all stakeholders involved in an engineering development process. They provide a good means to analyse the knowledge domain, allowing to separate the descriptive and the problem–solving knowledge. They can also be as generic as needed allowing its reuse and easy extension. These features made ontologies useful for representing the knowledge of software engineering techniques applied to autonomous systems. This work describes an ontology–based framework consisting of two intertwined elements: a domain ontology for autonomous systems (OASys) to capture any autonomous system's structure, function, and behaviour; and an ontology–based engineering methodology that generates models for autonomous systems, based on the knowledge contained in OASys and other domain ontologies. Both elements have been used in a case study to assess the suitability of the developed framework.

Keywords: Ontology, ontology-based methodology, knowledge-based engineering, autonomous systems.

1 Introduction

Autonomous systems refer to systems capable of operating in a real-world environment without any form of external control for extended periods of time. Reasons to provide systems with autonomy range from cost reduction to improved performance and dependability. Moreover, managing the increasing complexity of these systems has turned into delegating their configuration, optimisation, and repair to the systems themselves [1]. Autonomy is understood as a mongrel property that requires a combination of different capabilities:

- To obtain data and information (either from the environment or the system itself)
- To transform or refine information in a way that can be used by the decision–making elements
- To handle, to understand and to generate concepts

R. Setchi et al. (Eds.): KES 2010, Part I, LNAI 6276, pp. 522–531, 2010.

- To leverage cognitive aspects, meaning by such the capacity to reason, to infer and to learn
- To make decisions to achieve the system's goal, based on data provided at a sufficient level of detail
- To disseminate the decision taken to the appropriate execution or action elements

The former aspects require new engineering paradigms to cater for the idiosyncrasy of such systems: the perception process as major input for the system's knowledge, a more precise definition of system's goals to allow some decision–making to be moved from the designer to the system itself, the potential reconfiguration of the system as the goals, capabilities or the environment changes, and finally the means to assess the adequacy of the current configuration against the current mission. We are carrying out a long–term research programme, considering a wide range of autonomous systems. The strategy to increase a system's autonomy will be by exploiting cognitive control loops based on knowledge in the form of different models: of the system, of the environment, and of the task the system must fulfil in a particular environment. We have followed an ontological approach to capture the concepts of our research programme, as a knowledge–representation and software support for autonomous system's engineering.

This paper describes our efforts and conclusions on applying such approach, with the following structure. Section 2 briefly describes our ontology–based framework, consisting of a domain ontology for autonomous systems (OASys), and a related methodology to exploit OASys to generate models based on the knowledge it contains. Next, Section 3 exemplifies the usage of the framework in a research testbed. Section 4 reviews previous research on the domain, and compares it with our approach. Finally, Section 5 provides some concluding remarks, and suggests further work.

2 OASys Framework for Autonomous Systems

The OASys Framework captures and exploits the concepts to support the description and the engineering process of any software–intensive autonomous system. This has been done by developing two different elements: an autonomous systems domain ontology (OASys), and an OASys-based engineering methodology.

2.1 Ontology for Autonomous Systems (OASys)

OASys is a domain–ontology consisting of two ontologies (Fig. 1): the ASys Ontology with the ontological elements to describe an autonomous system structure, behaviour and function, and the ASys Engineering Ontology to provide the concepts for the engineering process of an autonomous system. Each one is organised using Subontologies, and Packages. Subontologies address autonomous systems' description and engineering at different levels of abstraction, whereas packages are organisational elements used to gather concepts semantically related to a specific aspect within a subontology. The different packages have been

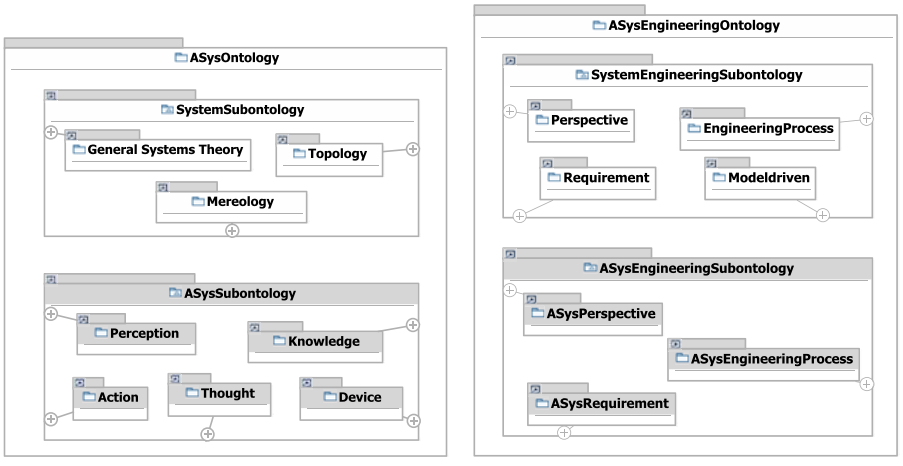

Fig. 1. OASys structure: ontologies, subontologies and packages

formalised using UML class diagrams to show the ontological elements in terms of concepts, attributes and relationships.

System Subontology contains the elements necessary to define any system, organised in three different packages: the *General Systems Theory Package* gathers concepts to characterise any kind of system's structure and behaviour based on the General Systems Theory; the *Mereology Package* collects taxonomies of the whole–part concepts and relationships based on existing mereological theories; the *Topology Package* provides concepts for topological connections based on topological concepts.

ASys Subontology specialises the previous concepts for autonomous systems, containing several packages to address its structure, behaviour and function: the *Device Package* gathers concepts to describe the different aspects of devices; the *Perception Package* to formalise the perceptive and sensing processes; the *Knowledge Package* to conceptualise the different kinds of knowledge used, such as the types of goals in an autonomous system; the *Thought Package* to describe the goal–oriented processes concepts; the *Action Package* summarises the concepts about the operations and performing actors.

System Engineering Subontology gathers the concepts related to an engineering process as general as possible, based on different metamodels, specifications and glossaries used in software–based developments. The subontology is organised as follows: the *Requirement Package* conceptualises system's requirements; the *Perspective Package* addresses viewpoints in a system development; the *Engineering Process Package* describes an engineering development process in terms of phases, tasks, and the obtained workproducts; the *Model–driven Package* provides the elements based on model–driven engineering.

ASys Engineering Subontology contains the specialisation and additional ontological elements to describe an autonomous system's generic engineering process, organised as different packages: the *ASys Requirement Package* to provide those concepts to characterise process and system quality requirements; the *ASys Perspective Package* to describe an autonomous system from different aspects; the *ASys Engineering Process Package* to describe an autonomous system generic engineering process.

2.2 The OASys–Based Methodology

The OASys–based methodology provides some guidelines and exemplifies the application of the OASys ontological elements to obtain the autonomous system's models during the phases of an engineering process. The ontological elements

Table 1. OASys–based Methodology: ASys Requirement phase

PHAS E	TASK	SUBTASK	WORK PRODUCT		RELATED OASys PACKAGE
ASys Requirement	System UseCase	UseCase Modelling	UseCase Model	System UseCase Model Subsystem UseCase Model	System Engineering Subontology: Requirement Package ASys Engineering Subontology: ASys Requirement Package
		Use Case Detailing	UseCase Specification	UseCase Description	
	Requirement Characterisation	Non-functional Requirement Functional Requirement	Requirement Specification	Process Characterisation System Characterisation	

Table 2. OASys–based Methodology: ASys Analysis phase

PHAS E	TASK	SUBTASK	WORK PRODUCT		RELATED OASys PACKAGE
ASys Analysis	Structural Analysis	System Modelling	Structural Model	Structure Model Topology Model	System Subontology: General Systems Theory Package, Mereology Package, Topology Package
		Knowledge Modelling	Knowledge Model	GoalStructure Model Procedure Model Quantity Model Ontology Model	ASys Subontology: Knowledge Package
	Behavioural Analysis	Behaviour Modelling	Behavioural Model	Behaviour Model	System Subontology: GST Package
	Functional Analysis	Function Modelling	Functional Model	Agent Model Operation Model Responsibility Model	ASys Subontology: Action Package ASys Subontology: Thought Package ASys Subontology: Perception Package

defined in the System Engineering and ASys Engineering subontologies serve as semantic guide for the phases, tasks, and work products names and usage. The methodology focuses on two main phases: the ASys Requirement phase to identify the autonomous system's requirements (Table 1), and the ASys Analysis phase to consider the autonomous system's analysis of its structure, behaviour and function (Table 2).

3 A Case Study: The Robot Control Testbed

The OASys Framework is being applied in the description and engineering of a Robot Control Testbed (RCT), which is a robotic–based application, consisting of a base platform and different interconnected subsystems to cover a wide range of functionalities. Not all the developed models are described in this section, only some to exemplify the usage of the ontology and the methodology.

The RCT ASys Requirement Phase identified stakeholders' requirements, using traditional engineering techniques. An Use Case Model as WorkProduct was obtained, considering the elements in the Requirement Package: the concepts of *Subject*, *UseCase*, and *UseCaseActor* and the relations of *include* and *appliedTo*. Figure 2 shows how these elements are instantiated into the case study ones. The top white UML classes represents the original concepts and relationships in the Requirement Package. The lower shaded classes, the instantiated elements. Each instantiated class is related to its corresponding one using a UML generalisation relation to express the is–a relationship between them. The instantiated relation receives the same name as the original relation in the package, known as construct overloading. However, the specialised relation is so with a different range and domain, and thus with different semantics.

As part of the ASys Analysis Phase, the Structural Analysis task analyses the autonomous system's structure using the structural concepts General Systems Theory Package, and the mereotopological relationships in the Mereology

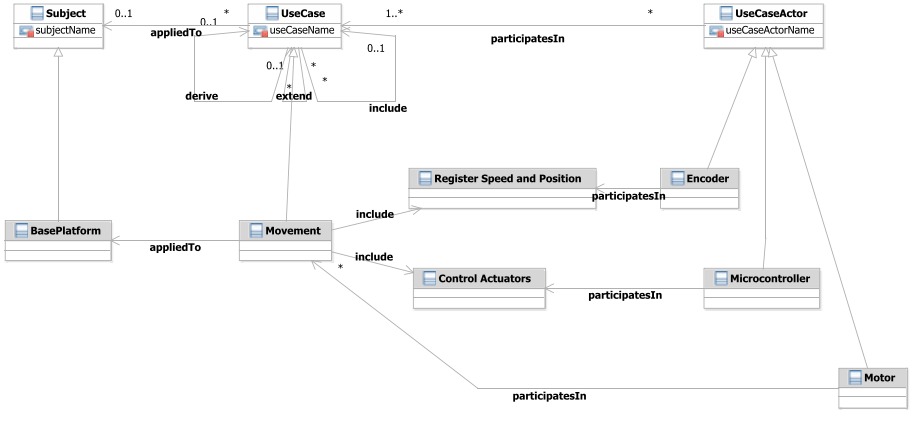

Fig. 2. Use Case Modelling: example of instantiation for the Base Platform of the RCT

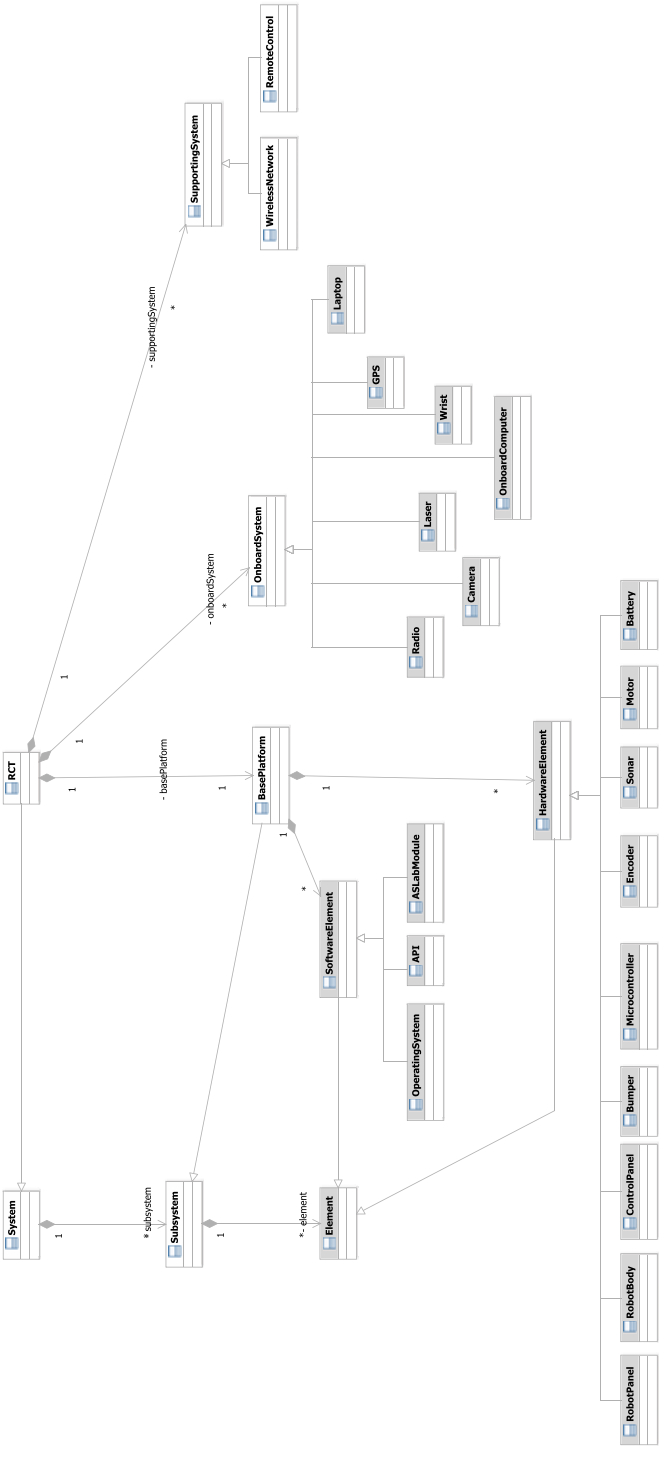

Fig. 3. System Modelling subtask: the RCT Structure Model

and Topology Packages. As WorkProduct, a Structural Model for the RCT is obtained instantiating these ontological elements, consisting of a RCT Structure Model and a RCT Topology Model. Figure 3 shows an example of the Structure Model for the RCT. On the left, the original UML classes that formalise the concepts of *System, Subsystem*, and *Element*. On the right, their instantiation into RCT actual elements. To point out that part–whole relations defined explicitly in the Mereology Package, such as *isPartOf* or *isExclusivelyPartOf*, can be instantiated using the UML aggregation and composition relations. The UML composition relation has been used to model that the RCT System is composed of a Base Platform, several Onboard Systems and different Supporting System. The original UML relation cardinalities on the left side of the figure have been particularised into the required ones in the RCT. For example, the RCT only contains one Base Platform, which is expressed by the cardinality 1 close to the UML class BasePlatform. Subsystems can contain different subsystems of elements, hence they can be in turn decomposed. In the RCT System Model, the Base Platform is decomposed into the software and hardware elements, using respectively the SoftwareElement and HardwareElement classes. UML Generalisation relations are used to classify additional Elements within the two former categories.

4 Discussion and Related Work

Our ontology–based framework merges two different aspects. Firstly, it is used to describe autonomous systems. Secondly, it should provide support for conceptual modelling and software engineering for such systems. Regarding autonomous systems, ontologies have been used with an approach closer to the viewpoint of knowledge representation based on a specification of a conceptualization, as opposed to defining the meaning of terms or to understanding the world. Ontologies are used as representation–based mechanisms based on a computational language, to describe the different entities participating in the design and operation of the autonomous systems: different domains, the environment, the objects the systems interact with, the possible actions to be taken, resources to be considered, etc. In a similar way, OASys has conceptualised how the engineers developing autonomous systems characterise them, describing the different elements taking part in their operation. However, our approach differs in considering these elements not only in terms of environment objects, resources or actions, but on the main aspects we consider fundamental for autonomy: the perception process, the knowledge to be used, the system's goals, the thought process as functional decomposition, and the actors to carry out the system's actions.

Moreover, we have considered the autonomous systems domain with a global view. Previous ontologies developed for this kind of systems focused on a particular application type, such as mobile robots or agent based systems. Our approach has been to define the different elements to describe any autonomous system, in a way general enough to be reused among different applications. OASys can be further complemented with subdomain, task or application ontologies, without loosing its reusability and generality features.

OASys has provided similar benefits to those referred to in the literature on ontologies for autonomous systems [10]: clarifying the structure of knowledge, knowledge sharing and reuse, and easing the interoperability among heterogeneous agents. Additional advantages have been a common conceptualisation not only of the autonomous systems domain but also of the engineering process for this kind of systems, making easier sharing the terminology among different developers. It remains to explore the reasoning capabilities based on the ontological relationships and commitments defined in OASys.

For software engineering, ontologies have played a twofold role [7]: as conceptual basis for the definition of software components in the software engineering process, and to describe the terminology of the software engineering domain itself. OASys and its methodology fall within the first role, providing a conceptual basis to define the early stages of an engineering process specific for autonomous systems according to our research view. Ontologies in this domain provide a series of benefits [9], such as a specialised representation vocabulary, transference of knowledge in software projects, same conceptualisation for different software applications, and conceptual mismatches reduction among users. Similar benefits have been obtained during the testbeds development using OASys: specific vocabulary for our research programme, same conceptualisation for software applications to be developed, and common meaning of elements throughout the engineering process.

5 Concluding Remarks and Further Work

We learned some lessons whilst developing the models for the case study. The model development benefited from the underlying ontological commitments in OASys, avoiding meaning and conceptual mismatches. However, the ontology development and the construction of the models was not exempt of challenges.

The decision to formalise OASys using a software engineering language such as UML was based on two facts. Firstly, the review of software engineering techniques and ontologies made as part of the research showed most of them as being UML–centered. Secondly, OASys was designed to support the model–driven engineering process in the research programme. UML is not an ontology development language, hence it has been considered to have some drawbacks for this task [5]: lightweight ontologies, incomplete semantics, software heritage, and lack of inference mechanisms. It has as advantage its widespread use among engineering practitioners, the support of a graphical representation, and the existence of UML–based tools. Some of the shortcomings have been addressed in the Ontology UML Profile [4], and concretise in the Ontology Definition Metamodel (ODM) [8] that defines metamodeles, mappings and profiles to allow the interaction between UML and ontological languages such as OWL and RDF.

An additional issue was to define the role of OASys for metamodeling. The case study models were obtained by instantiating, i.e., creating an entity that conforms to the definition. During the instantiation in the case study, we encountered the two forms of metamodelling described in [3], [2]: linguistic instance–of

relationships used to express that the concept has been made from a specification concept (e.g., to describe specific autonomous system's objects), and ontological instance–of relationships (is–a) used to express its similarity to an ontological element of a higher level ontology (e.g. to instantiate the packages ontological elements in the case study elements). The existence of the two different metamodelling approaches leads to some metamodelling problems [6]. To address the ontological and linguistic instantiations, OASys currently plays the role of an ontological model expressed with UML modelling constructs, inheriting its features from the UML Metamodel itself.

The models in this paper explicitly show the ontological is–a relationships, as the UML generalisation relations between the OASys elements and the actual RCT ones. This is time consuming and clutters the models. This semantic information is not shown when obtaining the models using an UML based tool, where the relation is inherited but not explicitly shown. We are evaluating the use of roles in the UML relations to express the original element one class is instantiated from. UML stereotypes are also considered to address this point.

Finally, the construction of the conceptual models was made ad–hoc, i.e., specifically for the case study. A proper methodology based on ontological and software patterns with the aid of conceptual modelling tools is a further step to enrich these conceptual models. This methodology, to be defined in the next stage of our research programme, will define how a concept is selected, how to integrate a concept into a pattern, how to establish and to import its relationships with other concepts, and how to detail or to add its attributes as part of the development of a concrete model.

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
2. Assmann, U., Zschaler, S., Wagner, G.: Ontologies, meta–models, and the model–driven paradigm. In: Calero, C., Ruiz, F., Piattini, M. (eds.) Ontologies for Software Engineering and Software Technology, pp. 249–273. Springer, Heidelberg (2006)
3. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE Software 20(5), 36–41 (2003)
4. Gasevic, D., Djuric, D., Devedzic, V.: Model Driven Architecture and Ontology Development. Springer, Heidelberg (2006)
5. Gómez Pérez, A., Fernández López, M., Corcho, O.: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. In: Advanced Information and Knowledge Processing. Springer, Heidelberg (2004)
6. Henderson-Sellers, B., Gonzalez-Peres, C.: Metamodelling for Software Engineering. John Wiley and Sons Ltd., Chichester (2008)
7. Hesse, W.: Ontologies in the software engineering process. In: Tagungsband Workshop on Enterprise Application Integration (EAI 2005), Berlin, Germany. GITO–Verlag (2005)

8. Object Management Group. Ontology Definition Metamodel Version 1.0 (May 2009)
9. Ruiz, F., Hilera, J.: Using ontologies in software engineering and technology. In: Calero, C., Ruiz, F., Piattini, M. (eds.) Ontologies for Software Engineering and Software Technology, pp. 49–95. Springer, Heidelberg (2006)
10. Stojanovic, L., Schneider, J., Maedche, A., Libischer, S., Studer, R., Lumpp, T., Abecker, A., Breiter, G., Dinger, J.: The role of ontologies in autonomic computing systems. IBM Systems Journal 43(3), 598–616 (2004)