# Model-based Engineering of Autonomous Systems using Ontologies and Metamodels

Julita Bermejo-Alonso Carlos Hernández Ricardo Sanz

.

## I. Introduction

There are two strategies to address engineering problems: 1) focus on a specific application and devise a clever solution for it or 2) try to develop generic solutions that can be applied by tailoring to specific applications. The first approach is usually much more effective and fast when building a single system; the second, however, is less effective for a single application and much more time consuming. This being the case, the generic approach offers two effects that make it desirable: reduces the cost of developing several systems and, most interesting, provides theoretical insights into the nature of the engineering problem. This is the kind of research described in this paper, where we develop a generic approach to the engineering of autonomous systems and apply it to the specific case of a fault-adaptive robotic application.

The growing complexity of missions and behaviours demanded to autonomous systems imply the use of a precise, yet increasingly complex, knowledge shared by the different stakeholders in the process. For an autonomous system to behave appropriately in an uncertain environment, the system must have some kind of meaningful representation of what it perceives as it observes entities, events, and situations in the world. It would also be desirable for the system to have an internal model that captures what it knows and learns, as well as a mechanism to compute values and priorities that enables it to decide what objectives and goals to fulfil. Knowledge is not only the cornerstone of intelligent artificial systems, such as autonomous robots, it lies at the centre of the engineering of such systems. This is true not only for intelligent entities but also for all kinds of autonomous systems.

Knowledge about the domain is usually acquired by analysing system implementations and talking to experts. This knowledge constitutes the very engineering resource when designing a system, to fulfil users' requirements. Nevertheless, the development process could become a daunting task due to two major issues [1]:

- Lack of a global picture of the problem: despite a thor- ough research to understand the problem space, systems developers encounter a myriad of different viewpoints, partitioned domains of expertise, and incompatible prior- ities.
- Never-ending changes throughout development: it is not uncommon that research and development projects should be updated, changed or reconsidered whilst the engineering efforts take place.

To overcome them, our research aims at capturing the core concepts and relationships in the *autonomous systems domain*, to enable the systematisation of knowledge, and its use in the model-based engineering of autonomous systems. Engineers have always built models: to describe physical systems, to specify products to be built, and to describe system interaction with the environment. Models serve as the basis for analysis and design in an engineering process.

Nevertheless, models need a precise meaning to allow for consistency, sharing, traceability and reasoning. This is where our proposal based on ontologies fits in: an ontological analysis of a domain becomes an excellent enabler for modelling, since it allows to identify the terminology needed to model a domain as well as establishing the meaning of the concepts and their relationships.

In this paper, we propose the use of an ontological approach to engineer autonomous systems, as an ontology-driven engi-

neering framework that produces models of the system as core assets. This technology has been applied to a mobile robot, where models can be used throughout the robot life-cycle, from analysis through design up to its implementation and maintenance, as well as during self-operation.

The rest of the paper is organised as follows: Section II presents a summary of previous efforts related to our research. Section III situates the background for the research, the *ASLab ASys Project*, and the ontology-driven engineering framework. Section IV explains the ideas on how to develop a metacontrol for autonomous robots, with Section V describing the engineering application of the methodology and the metacontrol. Finally, Section VI concludes the paper with some insights and conclusions.

## II. Related Research

Knowledge engineering faces some theoretical challenges to unleash the full potential of a model-based approach for the engineering of autonomous robots. Following, we review the literature on the issues that we have addressed in this work.

### A. On Ontologies, Models and Metamodels

A model is an abstraction of a system leaving aside irrelevant details hence focusing on the relevant ones, thus allowing significative predictions or inferences to be made [2]. In software engineering a model is an artifact constructed according to a certain modelling language that describes the system, including usually a graphical representation of it using different types of diagrams.

A metamodel is a specification model for a system, where each system happens to be itself a valid model expressed in a concrete modelling language. In other words, a metamodel is a prescriptive model of a modelling language [3], defining explicitly the constructs and rules needed to build up specific models within a domain of interest. A model thus conforms to its metamodel.

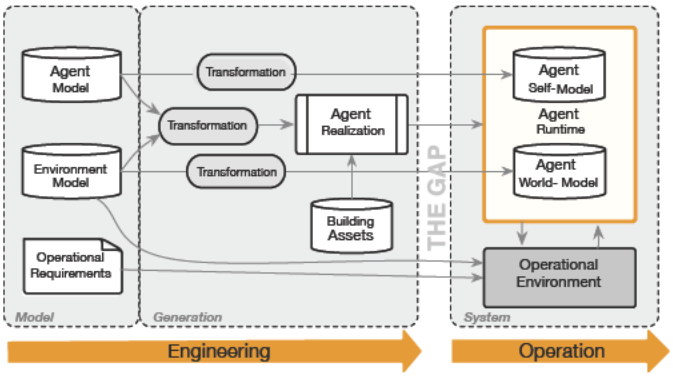


Fig. 1. The Engineering/Runtime Gap

A special flavour of models, called ontologies, became relevant to the knowledge engineering community as a mean to represent knowledge to support intelligent behaviour [4], where ontologies are typically intended for exploitation of knowledge at runtime, and must be "formal" in the sense of being understandable by a computer. An ontology is a formal, explicit specification of a shared conceptualisation, as a machine-readable abstract model where relevant concepts and relationships are identified and explicitly defined by consensus in a group [5]. Ontologies are used to model domains in knowledge-based systems. Models are formalized using modelling languages. Ontologies expressed in that very same modelling language can serve to build those models, acting as analysis metamodels defining all valid models for a domain [6], [7].

### B. On Ontology-driven Engineering

Ontologies facilitate good modelling as analysis models describing the conceptualization of a domain: its terminology, their definitions, and relationships. All these elements can be reused across multiple engineering domains to model real world applications. Ontologies could be regarded as reusable building components when modelling systems at a knowledge level, since they enable knowledge sharing and reuse. Following this approach, ontologies are used as the backbone in Model-Based Systems Engineering (MBSE) and software development [8]. As an example, the Ontology Action Team [9], a part of the INCOSE MBSE initiative, whose goal is to consider ontologies as a component in engineering modelling activities.

Ontologies also serve as representational mechanisms based on a computational language, to clarify and share the domain knowledge, providing a common representation vocabulary for software engineering processes, helping to transfer knowledge as well as to simplify the development cycle from project to project. The ontological analysis clarifies the structure of knowledge used by the practitioners in the domain, easing its sharing and reuse among them.

When it comes to the particular domain of autonomous robots, ontologies have been applied for autonomous robots description and operation, as knowledge representation used to characterize the autonomous robots domain, the tasks to perform or the environment where the autonomous robot is placed in, or the knowledge base used by the robot for autonomous operation [10]. An IEEE standard for robotics and automation has been recently approved, where ontologies are used to represent knowledge in both domains [11], to allow for unambiguous knowledge transfer among any group of human, robots and artificial systems [12], [13].

## III. Our Research

### A. Systems Engineering in the ASLab ASys project

The ASLab ASys Project tries to develop a generic autonomous control system architecture and associated reusable assets as well as an engineering methodology using this architecture and assets to build autonomous systems. A paradigmatic example of the autonomous systems we target are mobile robots that shall perform a mission coping with both unstructured environments and inner faults. The central methodological idea that drives this research is the use of a

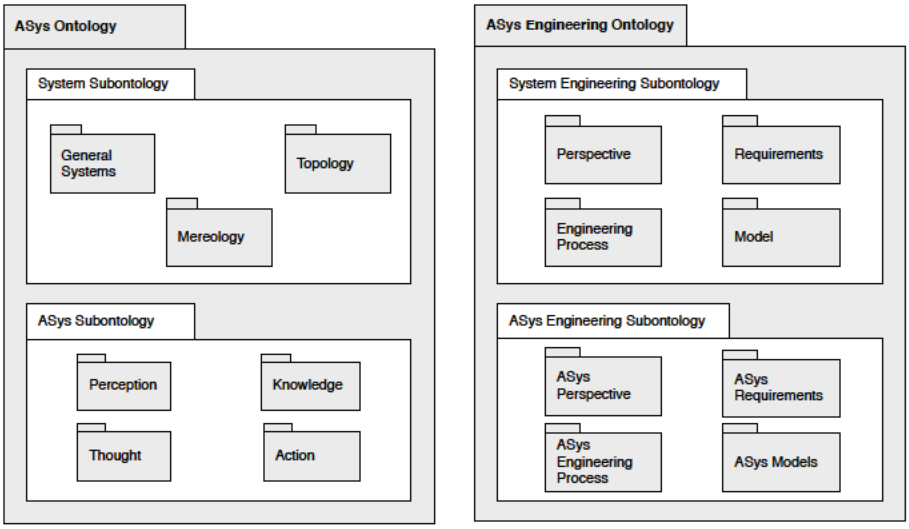


Fig. 2. OASys Structure: ontologies, subontologies, and packages.

full life-cycle model-based approach where system models are used to build the robot and also to enable its fault handling capabilities through meta-cognitive capabilities based on those system models[1].

The system model that is the cornerstone of this approach shall ease both the construction and the operation of autonomous systems (ASys). This model is integrated and executable helping bridge the gap between the engineering and operation phases of the system life-cycle. Both the system architecture and the engineering methodology orbit around these ASys models: models are core assets of the methodology and models are used at runtime by the autonomous system to drive its self-x mechanisms [14], [15] to increase its autonomy.

In this vision of breaking the engineering/runtime gap (Fig. 1), the sharing of concepts between the engineers and the metacognitive engine of the robot is critical. This is the reason why ontologies and metamodels are a crucial aspect of this approach.

### B. The Ontology-driven Framework

The developed framework is ontologically driven, consisting of two elements. The first one is a *domain ontology* to capture the system structure, function and behaviour of autonomous systems. The second element is an *ontology-driven engineering methodology* to develop specific autonomous systems — e.g. mobile robots. This methodology is based on MBSE and uses models of the system as core assets. These models can be used throughout the whole engineering life-cycle, from design through implementation, up to validation and maintenance, and most importantly, to runtime to drive self-x mechanisms.

*1) The domain ontology — OASys:* The **Ontology for Autonomous Systems** (OASys) is a two-layered dual ontology [16] (see Fig. 2). The two layers address different levels of abstraction: a higher level of ontological elements addressing *general systems*; and a lower level specifically addressing *autonomous systems* (ASys). The two dual ontologies describe the *systems themselves* (ASys Ontology) and the *engineering process* (ASys Engineering Ontology). Each ontology is internally organized in subontologies and packages, to allow for reusability and future extensions. As a domain ontology, OASys is a special kind of ontological model expressed using a modelling language (UML), used to define the ontological elements and their relationships (UML class diagrams).

*2) The methodology — ODEM:* The **OASys-driven Engineering Methodology** (ODEM) provides support for ontology-based autonomous systems development based on the OASys ontological elements [17]. The methodological elements of ODEM are (see Fig. 3):

- An autonomous system engineering process as a combination of phases (e.g. Requirements phase, Analysis phase, Design phase, etc.) and related tasks for each phase (e.g. System Use Cases, Requirements Characterization, Structural Analysis, Behavioural Analysis, Functional Analysis, etc.).
- The work products to be obtained as a model kind that could consist of different diagrams and specifications (e.g. Structural Model as a combination of Structure and Topology models from the Structural Analysis task).
- The specification of the related OASys ontologies, packages and elements to be used in the work products, e.g. Requirement Package and ASys Requirement Packages for the Requirements phase.

To obtain the system models as work products, the ontological elements in OASys are instantiated or refined as specified in ODEM, considering two forms of types of metamodelling relations: linguistic (instance-of relation) and ontological (is-a relation) [19].

[1]To be precise, runtime models derived from them. See Fig. 1.

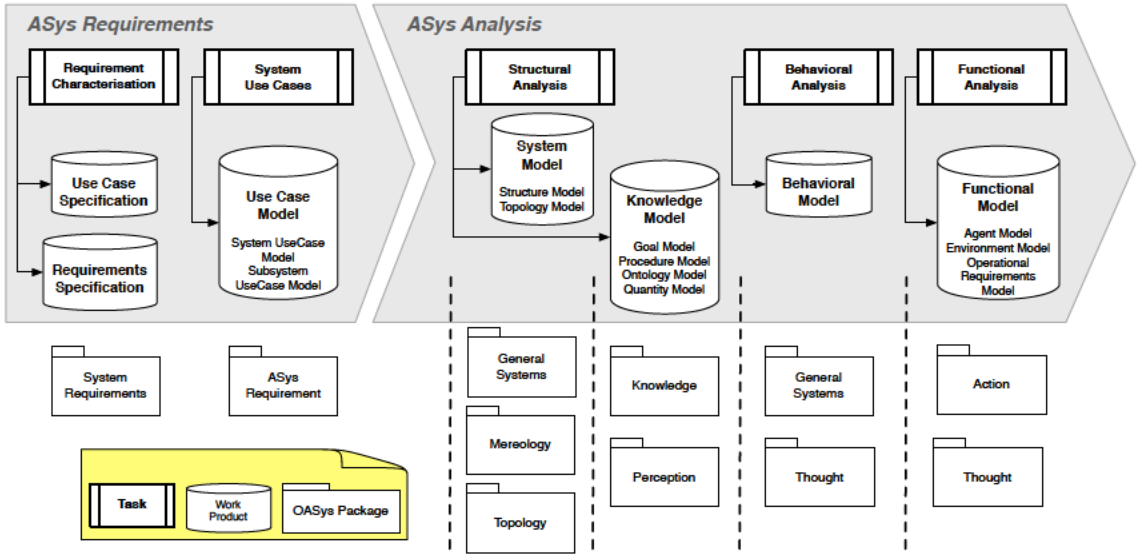


Fig. 3. ODEM Phases, Work Products and related OASys Packages

## IV. Model-based Metacontrol

This ontology-driven framework has been applied to the ASys idea of bridging the engineering/runtime gap in a mobile robot application. The **Operative Mind** (OM) architectural framework [20] defines a reference control architecture that exploits the knowledge in the engineering model of the autonomous system to implement self-x mechanisms: self-awareness, self-monitoring and self-repairing.

The core elements of the OM metacontrol architecture are: 1) a model of the autonomous system, and 2) a metacontrol module that uses such model. The metacontroller can be integrated on top of any component-based control system. The model is a functional model that relates the system structure and configuration of components to the functions designed in the system to fulfil its mission objectives. The specification of the metacontroller as a reference architecture makes our solution applicable to all robotic domains, and its model-based character allows for the re-usability of implemented metacontrollers for different applications.

The OM Architectural Framework also includes an engineering process defined with ODEM to build the metacontrol architecture in new autonomous systems, or integrating it into extant ones.

### A. TOMASys Metamodel

We have followed the metamodeling approach to define the functional model in OM. The Deep Model Reflection Pattern [21], is our solution for the capture of the meta control knowledge. For the knowledge to be executable at runtime as an explicit model, and that model being generated from the engineering model of the system, it shall conform to a metamodel from which a transformation exists from the engineering modelling language.

This metamodel, TOMASys, has been domain-focalised from OASys, and integrates concepts from other models and specifications [22], [23], [24]. TOMASys refines OASys concepts to account for the structure and functions in component-based robot control architectures. Fig. 4 shows the main elements in the TOMASys metamodel and the OASys concepts it refines.

### B. Metacontrol operation

The metacontrol module uses the TOMASys model of the autonomous system to monitor that its behavior fulfils the mission requirements. To do so, it closes a control loop whose reference goal is the TOMASys *objectives hierarchy* of the system. The metacontrol receives the monitoring information –the state of the system's components– through introspection probes, and acts on the system by reconfiguring it –if required– to adapt to behavioural divergences. It may activate or deactivate components, re-connect them or change their parameters as appropriate.

The metacontrol architecture consists of two loops that use the TOMASys model at the structural and the functional levels. At the lower loop, the structural and behavioural knowledge captured in the model *components classes* is used to update the instantaneous state of the components from the probes' readings.

This state is the input for the upper loop to infer the functional state of the system up to the system's objectives achievement, using the knowledge in the *functions* and their *require relationships*. Any deviation at the functional level is addressed by the metacontroller looking for alternative functions in the model. If found, the configuration of the components is commanded to the lower loop, which executes the reconfiguration actions required to realize it.

### C. Metacontrol Engineering

The OM Engineering Process (OMEP) has been developed based on ODEM (see Fig. 5), to define the activities that

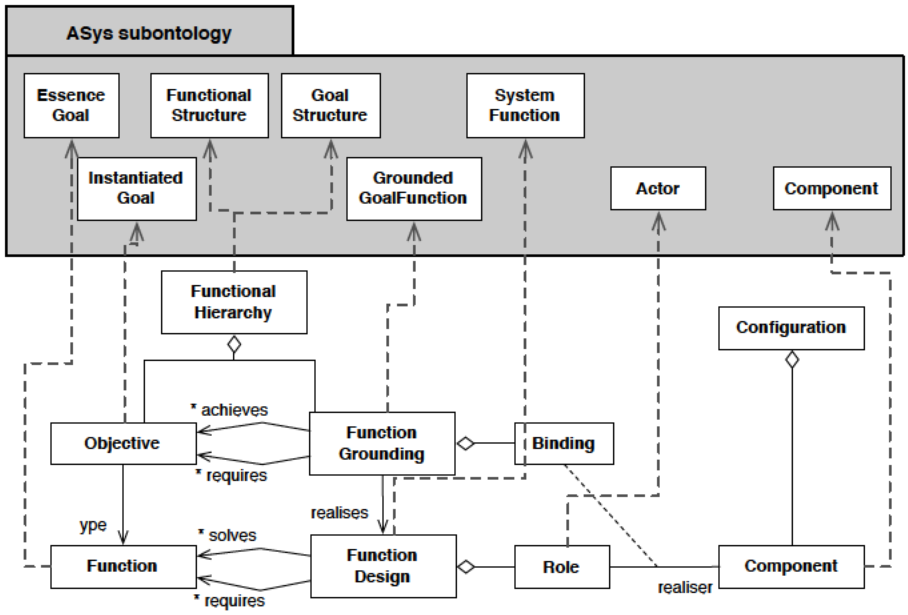


Fig. 4. Core elements of TOMASys and their refinement from OASys.

support the engineering of our metacontrol solution for autonomous applications. OMEP involves two subprocesses:

- The Domain Control Development subprocess uses ODEM in the analysis and design of the regular control of the autonomous system to obtain the modelling assets required for metacontrol.
- The Metacontrol Development subprocess has been greatly simplified thanks to the architectural and model-based approach and the application-independent assets that we have already developed. For a concrete robotic application, only the model and the integration of our meta control library for the specific platform of the robot must be developed, the later being reusable for any robotic system implemented with that platform.

## V. Fault-tolerant Robot Re-engineering

OMEP has been tested in the re-engineering of an autonomous mobile robot for patrolling. The goal was to enhance the robot with mission-level resilience to failures.

The robotic system consisted of a four differential wheeled base instrumented with odometry, a range laser and a Kinect camera for navigation. The control system was based upon the ROS navigation stack [25], with additional components developed for the patrol application. By adding an OM metacontroller, the robot would be able to recover to some extent from failures in its components, and thus continue navigation.

Firstly, in the ODEM ASys Requirements phase the metacontrol requirements were identified: the robot being able to recover operation in scenarios including a transitory failure in any of the control components and, more importantly, a permanent failure in the laser sensor.

Then, the ODEM ASys Analysis phase maps then into the Structural Analysis task (OASys *subsystems*) and the Functional Analysis task (OASys *operations*) following the metacontrol requirements. As result of the Structural Analysis task, a System Model consisting of Structure and Topology Models showing the component's configuration and their interconnection were obtained.

The Behavioural Analysis task produced the error models of the *components*. ROS components report errors and their criticality through a runtime log system. The Functional Analysis task identified the standard robot control architecture. Fig. 6 (a) and (b) show respectively, the navigation, localisation, motion, and sensing functions demanded from the robot according to the requirements, and how they are realised by the configuration of the components. The impact of the components' errors on the robot functionality was analysed empirically and captured in the TOMASys model during this task too. The standard metacontroller recovery for components provided the distinction for both laser errors. While a reset of the laser driver solved the transient failure without triggering a failure at the localisation function level, a permanent error in the laser (e.g. due to a hardware problem) scaled to an error in the localisation and navigation functions.

Next, the ODEM Functional Analysis task identified alternative *designs* for the localization and navigation *functions* that do not make use of the laser sensor. In this case, the scan information provided by the laser sensor, which is used both for localisation and obstacle avoidance during navigation, could be replaced by the Kinect 3D information, provided some additions, such as improved dead reckoning accuracy using a compass. The resulting Functional Model identified the roles of the components (OASys *actors*) required to realize the alternative robot control architecture. Fig. 7 (a) displays the alternative functions identified, with Fig. 7 (b) showing how them are realized by the final configuration of components.

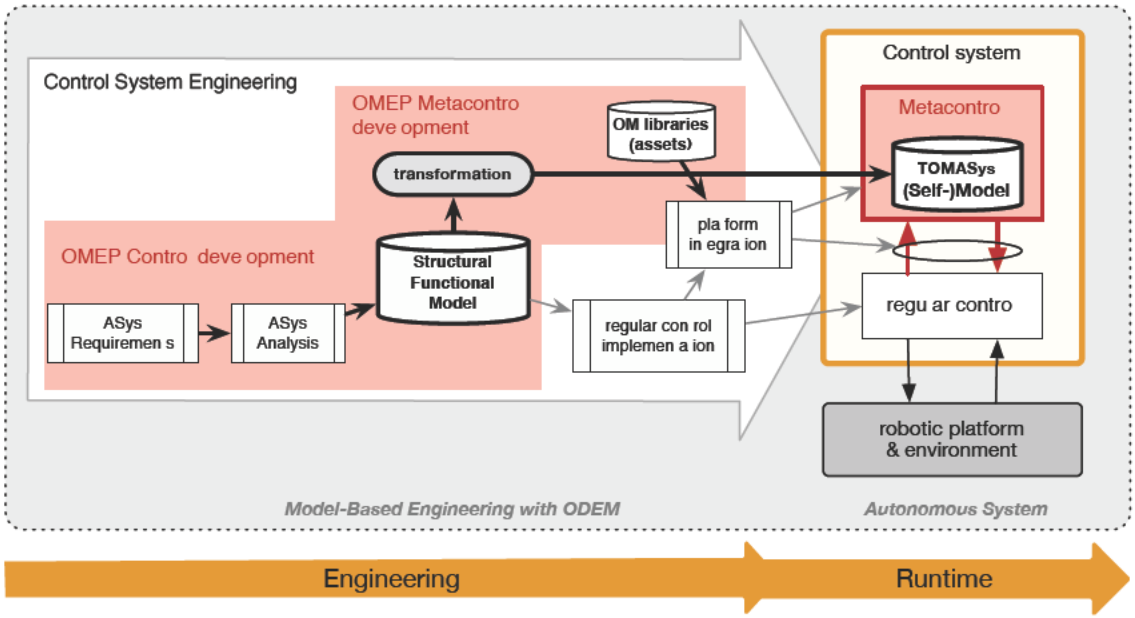


Fig. 5. Engineering with ODEM and OMEP.

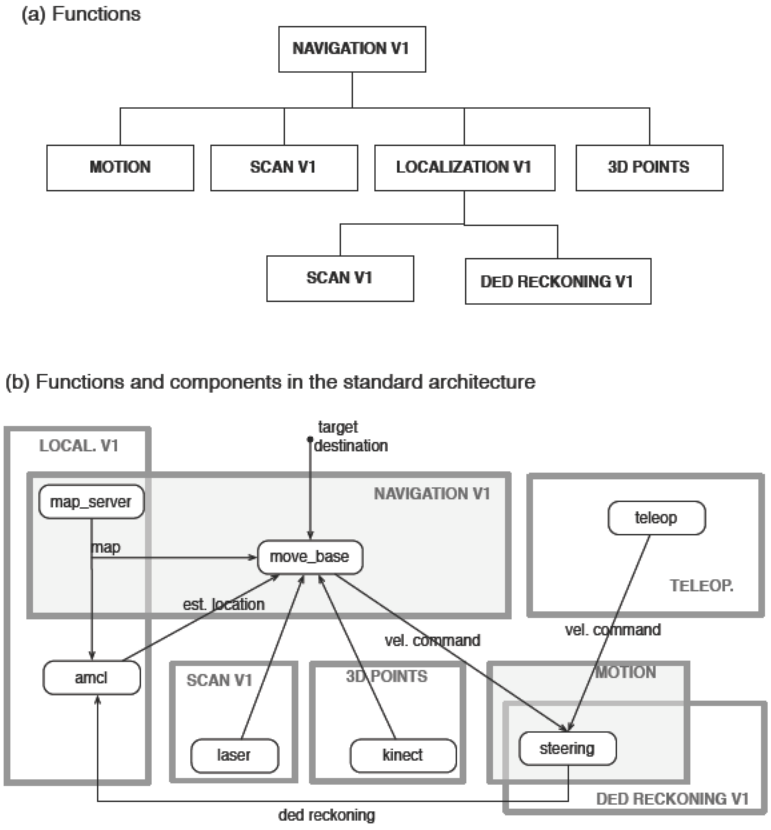


Fig. 6. Standard Robot Control Architecture.

Thanks to the models, the metacontroller can switch the robot control to the alternative architecture on the fly. Subsequently the re-engineering of the robot controller (regular control implementation in Fig. 5) only required the implementation of the new components, a Kalman filter to improve odometry and a module to map Kinect readings into laser-like scans. To implement the robot's metacontrol two application-independent libraries were built: a Java library with a multi-

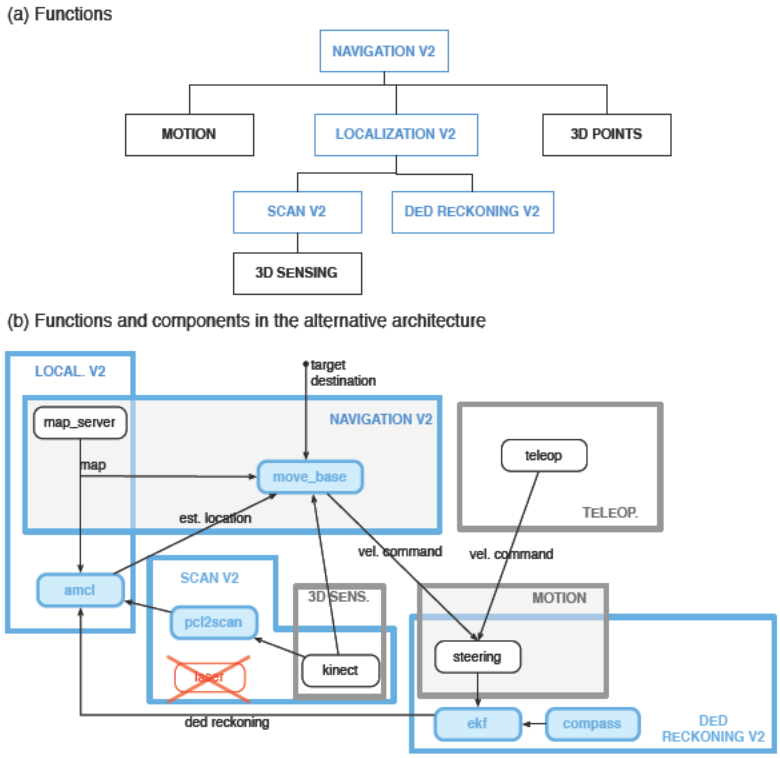


Fig. 7. Alternative Robot Control Architecture.

platform implementation of the metacontrol architecture; and a ROS package to integrate the Java metacontroller in any ROS-based system. This way, for the patrolling application, the Metacontrol Development subprocess only required the obtention of the TOMASys model of the mobile robot, taking as input the structural and functional models previously obtained in the ODEM Analysis phase.

While this solution may seem overkill to handle a sensor failure, the reader should consider the laser as a use case. The metacontrol architecture potentially provides for resilience to any failure, provided that sufficient engineering information is included in the model during the combined ODEM and OMEP processes.

For the sake of space, only some models of the application have been included here. For the interested reader, a comprehensive description of all the models in the application of ODEM and OMEP to the autonomous mobile robot can be found in [18], [20].

## VI. Conclusions

Autonomous systems engineering is hampered by the lack of systematic methods to incorporate cognitive capabilities to systems. AI-based autonomous systems employ sophisticated algorithms from software libraries but lack a established methodology and an architectural foundation. This is especially so when dealing with metacognitive capabilities, where system-specific idiosyncrasies permeate the knowledge and reasoning methods of the metalayer.

A more reusable and improvable methodology and assets for autonomous systems construction in a broad application domain is in serious need[2]. However, providing domain-wide solutions is not an easy task. Domain and multi-domain engineering requires deep conceptualizations and generic implementations to move from mere craftsmanship into true systems engineering for the autonomous systems domain that we are addressing.

Ontologies are an excellent mechanism to conceptualise the knowledge of a domain, providing the structured knowledge required to describe and to engineer the autonomous system — e.g. a mobile robot or a whole factory. Ontologies as enablers for modelling, also allow to obtain models for the description and the engineering of autonomous systems.

Additionally, formalised ontologies develop into exercisable models that can be used as core assets fuelling the whole model-based systems engineering process. Concepts from the mind of the autonomy engineers are reified into pieces that are used to build the autonomous system. This streamlines the engineering process, effectively breaking the gap between engineering-time and run-time. This is a necessary step because future systems shall be autonomous beyond their mission responsibilities and be able to take care of themselves.

Our approach has produced a framework that combines a general ontology for autonomous systems (OASys) and a

[2]This is specially important nowadays due to the necessary integration of autonomous and non-autonomous supervised systems into socially-critical large-scale systems-of-systems.

methodology for ontology-based MBSE (ODEM). These are the two cornerstones of a model-based autonomous systems engineering strategy that addresses the broad domain of autonomous systems.

This ontology-driven framework has proven its value enabling the development of a more domain-specific architectural framework for building self-x mechanisms into autonomous robots. Functional concepts are used for representing the robot elements, being instantiated into a self-model that enables run-time reflection of the autonomous robot.

This self-model sits at the core of a reusable architecture for robot self-awareness (OM). Firstly, ODEM's theoretical modelling stand has provided the knowledge engineering guidelines and foundational concepts for the building the self, functional model used by the metacontroller. This is rendered using TOMASys, a functional and componental metamodel that has been domain-focalised [26] from OASys. Secondly, ODEM has allowed to specify the engineering activities to build the metacontrol solution in an autonomous robots application, identifying the key assets involved — hence producing the OM Engineering Process (OMEP). This domain-focused application — from general autonomous systems to self-aware robots — is not a trivial task given the meta-aspects involved. Note that by the use of OASys, the robot can think about its own realisation — as an engineer would do.

Obviously, all this process seems too complex for engineering a single, simple, fault-tolerant mobile robot controller. However the extremely general approach based on a general ontology for autonomous systems will pave the way to both i) easy extension of the robot metacontrol architecture to address a broader set of robot issues at the mission and meta-mission levels; and ii) portability of the self-awareness architectural framework from the robotics domain to other domains where run-time reflection would help keep the system working.

The strategy and assets described have the ambition of serving as basement for a wide autonomous systems engineering technology.

## REFERENCES


[1] A. Brown, J. Conallen, and D. Tropeano, *Model-Driven Software Development*. Springer-Verlag Berlin Heidelberg, 2005, ch. Introduction: Models, Modeling, and Model-Driven Architecture (MDA), pp. 1–16.

[2] T. Kuhne, "Matters of (meta-)modeling," *Software and Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.

[3] B. Henderson-Sellers, "Bridging metamodels and ontologies in software engineering," *Journal of Systems and Software*, vol. 84, no. 2, pp. 301–313, 2011.

[4] C. Atkinson, M. Gutheil, and K. Kiko, "On the relationship of ontologies and models," in *Meta-modelling and ontologies*, ser. Lecture Notes in Informatics. GI-Edition, 2006, vol. P-96, pp. 47–60.

[5] R. Studer, V. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *IEEE Transactions on Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.

[6] U. Assmann, S. Zschaler, and G. Wagner, "Ontologies, meta–models, and the model–driven paradigm," in *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz, and M. Piattini, Eds. Springer-Verlag Berlin Heidelberg, 2006, ch. 9, pp. 249–273.

[7] H. Graves and M. West, "Current state of ontology in engineering systems," 2012, http://www.omgwiki.org/.

[8] J. Pan, S. Staab, U. Assmann, J. Ebert, and Y. Zhao, *Ontology-driven Software Development*. Springer Heidelberg, 2013.

[9] OMG MBSE Initiative, "Ontology Action Team." [Online]. Available: http://www.omgwiki.org/MBSE/doku.php?id=mbse:ontology

[10] C. Schlenoff and E. Messina, "A robot ontology for urban search and rescue," in *Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems*. Budapest, Hungary: ACM Press, November 2005, pp. 27–34.

[11] ORA Working Group, "1872-2015 IEEE Standard Ontologies for Robotics and Automation," IEEE Standard., February 2015.

[12] L. Paull, G. Severac, G. Raffo, J. M. Angel, H. Boley, P. Durst, W. Gray, M. Habib, B. N. Nguyen, S. Ragan, S. Saeedi, R. Sanz, M. Seto, A. Stefanovski, M. Trentini, , and H. Li, "Towards an ontology for autonomous robots," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2015)*, Portugal, 2015, pp. 1359–1363.

[13] B. Bayat, J. Bermejo-Alonso, J. Carbonera, T. Facchinetti, S. Fiorini, P. Goncalves, V. . Jorge, M. Habib, A. Khamis, K. Melo, B. Nguyen, J. Olszewska, L. Paull, E. Prestes, V. Ragavan, S. Saeedi, R. Sanz, M. Seto, B. Spencer, A. Vosughi and H. Li "Requirements for building an ontology for autonomous robots," *Industrial Robot: An International Journal*, vol. 43, to appear, 2016.

[14] R. Sanz, "Envisioning conscious controllers," Keynote Speech in International Workshop on Software Systems, 2004.

[15] R. Sanz, I. López, J. Bermejo, R. Chinchilla, and R. Conde, "Self-X: The control within," in *Proceedings of the 16th IFAC World Congress*. Praga, Czech Republic: IFAC, 4-8 July 2005.

[16] J.Bermejo-Alonso, R. Sanz, M. Rodriguez, and C. Hernández, "Ontology engineering for the autonomous systems domain," in *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, ser. Communications in Computer and Information Science. Springer Berlin-Heidelberg, 2013, vol. 348, pp. 263–277.

[17] J. Bermejo-Alonso, R. Sanz, M. Rodríguez, and C. Hernández, "An ontology–based approach for autonomous systems' description and engineering: the OASys Framework," in *14th International Conference on Knowledge–Based and Intelligent Information and Engineering Systems (KES 2010)*, ser. LNAI, R. Setchi, I. Jordanov, R. J. Howlett, and L. C. Jain, Eds., vol. 6276. Cardiff, Wales, U.K.: Springer, Heidelberg, 2010, pp. 522–531.

[18] J. Bermejo-Alonso, "OASys: an ontology for autonomous system," PhD dissertation, E.T.S.I.I.M., Universidad Politécnica de Madrid, 2010.

[19] B. Henderson-Sellers, *On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages*. Springer, 2012.

[20] C. Hernández, "Model-based self-awareness patterns for autonomy," Ph.D. dissertation, Universidad Politécnica de Madrid, ETSII, Dpto. Automática, Ing. Electrónica e Informática Industrial, José Gutierrez Abascal 2, 28006 Madrid (SPAIN), October 2013.

[21] C. Hernández, J. Bermejo-Alonso, I. López, and R. Sanz, "Three patterns for autonomous robot control architecting," in *PATTERNS 2013, The Fifth International Conferences on Pervasive Patterns and Applications*, A. Zimmermann, Ed. IARIA, May 27 2013, pp. 44–51.

[22] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*. Springer-Verlag Berlin, 2006.

[23] OMG, *OMG Unified Modeling Language (OMG UML) Infrastructure Version 2.2*, Object Management Group, February 2009.

[24] OMG, "Robotic technology component specification v1.1," Object Management Group, Tech. Rep. formal/2008-04-04, April 2012.

[25] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: robust navigation in an indoor office environment," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 300–307.

[26] R. Sanz, I. Alarcón, M. J. Segarra, A. de Antonio, and J. A. Clavijo. Progressive domain focalization in intelligent control systems. *Control Engineering Practice*, 7(5):665–671, May 1999.