

---

# THE MECA PROJECT: USING AN OWL/RDF KNOWLEDGE BASE TO ENSURE DATA PORTABILITY FOR HUMAN SPACE MISSION OPERATIONS

L. Breebaart, A. Bos, T. Grant  
S&T BV  
Delft,  
The Netherlands  
breebaart@stcorp.nl

A. Olmedo Soler  
OK-Systems,  
Valencia,  
Spain

M. Neerinx, N. Smets,  
J. Lindenberg  
TNO-D&S  
Soesterberg,  
The Netherlands

U. Brauer  
Astrium ST,  
Bremen,  
Germany

M. Wolff  
ESA-ESTEC  
Noordwijk,  
The Netherlands

## ABSTRACT

*The Mission Execution Crew Assistant (MECA) project is an ESA research project that aims to boost the cognitive capacities of human-machine teams during planetary exploration missions in order to allow them to cope autonomously with unexpected, complex and potentially hazardous situations.*

*To achieve a high level of autonomy, the MECA software must have a semantically rich view of the world in which it operates. It must have an understanding of high level mission objects (such as goals, resources, actors, tasks, procedures, plans and schedules) but also of lower-level system information (such as vehicles, payloads, instruments, sensors, actuators, processors, telemetry, test results, and fault diagnoses).*

*In order to assist the human astronaut optimally, MECA must also be aware of human issues such as trust, task load, stress levels, and other cognitive aspects of human-computer interfacing. Following the general objective of human-machine partnership, a major part of the information should be easily shared (i.e. understood) by both human and machine actors.*

*For the MECA-2007 Proof of Concept demonstrator, the MECA team used a distributed RDF knowledge base, containing heterogeneous data described by various OWL ontologies. The common semantic worldview provided by the RDF data made it possible and even easy for the software to be aware of and reason about connections between concepts and data that are not normally considered interoperable. It was also instrumental in implementing parts of MECA as semantic web services, facilitating the sharing of knowledge among heterogeneous agents.*

## 1. INTRODUCTION

Over the past decade, there has been extensive debate about the needs for and expected benefits, complications and implementation of spacecraft autonomy. The flights of NASA's Deep Space One and ESA's PROBA spacecraft have been milestones in the development of autonomy. Drivers for autonomy are seen as interruptions and delays in the Earth-space communications link, spacecraft survival in the presence of on-board faults, and cost reduction in the ground segment. The consensus is that autonomy involves building additional functionality into the spacecraft.

Spacecraft autonomy brings with it complications for ground-based control. Controllers have all the usual problems of monitoring a distant system over a communications link with limited bandwidth, of assessing the on-board situation, of determining the appropriate course of action, and of generating and uplinking tele-

command-sequences. In addition, they have to contend with tracking and understanding the decisions taken by the spacecraft's autonomous systems and intervening where necessary (and possible).

For longer duration manned missions, such as the human exploration of the Moon or Mars, the concept of crew autonomy versus ground control is likely to undergo drastic changes. The crew has to be closely involved in the decision-making process and in resource management. For cost reasons, staffing for mission control will be reduced by comparison with present-day Shuttle and ISS missions. Crew motivation and satisfaction needs for long duration missions are additional rationales for their increased involvement in decision-making.

For day-to-day operations, the distance between the planetary explorers and Earth-bound mission controllers prohibits the controllers' involvement in real-time decisions. Mission control will take part in strategic and tactical planning and mission evaluation activities, leaving the crew responsible for execution-level planning and mission execution. To this end, the crew will need IT-based support.

## 1.1. THE MECA PROJECT

The main objective of the MECA project is to develop the user and software requirements and an architectural baseline for an IT-based system that will enable autonomous operation by human-machine teams during planetary exploration missions. The MECA system is seen as a distributed, personal ePartner in a ubiquitous computing environment (see Figure 1), designed to amplify the cognitive capabilities of human-machine teams to cope autonomously with unexpected, complex and potentially hazardous situations.

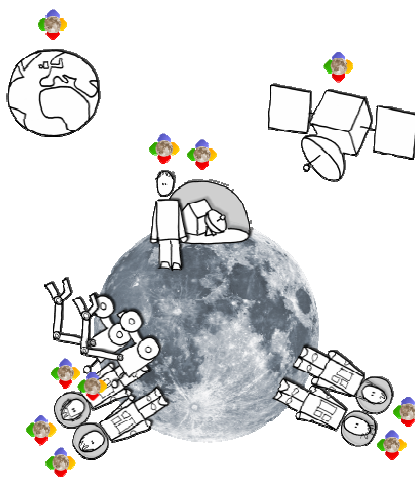


FIGURE 1. MECA AS UBIQUITOUS EPARTNER.

MECA is required to:

- Allow the crew to schedule and make decisions about short- and mid-term activities.
- Be pro-active with respect to health and status monitoring of equipment and facilities.
- Adjust monitoring and supervision algorithms as equipment and facilities evolve.
- Support collaboration for support teams and explorers separated in time and space [7].
- Provide “what-if” scenario evaluation.
- Provide crew with updated procedural knowledge [8].
- Provide crew with advice (potentially unsolicited) about possible alternatives and their effects on all available resources.
- Execute tasks delegated to it by the crew.

The MECA project is divided into several phases. An initial version of the user and software requirements have been defined in Phase 1, based on use-cases derived from four candidate mission scenarios and on a survey of research topics and emerging technologies. The requirements were refined in Phase 2 by defining an architectural baseline and by building and evaluating a proof-of-concept demonstrator of the MECA system using present-day technologies.

The MECA project began in August 2005. Phase 1 was completed with a System Requirements Review in February 2006. Phase 2 was completed with a final project report in December 2007. Phase 3 of MECA started in 2008 with a number of new follow-on projects (such as participation in MARS 500) intended to further refine the requirements baseline and bring the demonstrator software to a higher operational level.

An important part of the MECA architectural baseline is the Knowledge Base, a distributed repository for all shared information (models, data, events, etc.) that describe the mission world as the crew and the MECA software experience it. For the MECA proof-of-concept demonstrator, an ontology-driven OWL/RDF data store was used as a Knowledge Base implementation.

## 1.2. PURPOSE AND SCOPE OF THIS PAPER

The purpose of this paper is to report on the practical experiences of the MECA project in using OWL/RDF as an implementation vehicle for the MECA Knowledge Base in the proof-of-concept demonstrator software.

## 1.3. STRUCTURE OF THIS PAPER

This paper is divided into four sections. Section 1 gives the background, overviews the MECA project, and defines the purpose, scope, and structure of the paper. Section 2 describes the MECA architecture baseline, the Knowledge Base, and the associated data modelling. Section 3 discusses the experiences with and the advantages and disadvantages of the chosen OWL/RDF Knowledge Base implementation. Section **¡Error! No se encuentra el origen de la referencia.** summarises the work. Section 4 contains references.

# 2. THE MECA ARCHITECTURE AND KNOWLEDGE BASE

## 2.1. THE MECA ARCHITECTURE

The MECA Architecture, shown in Figure 2, is essentially a Service-Oriented Architecture (SOA).

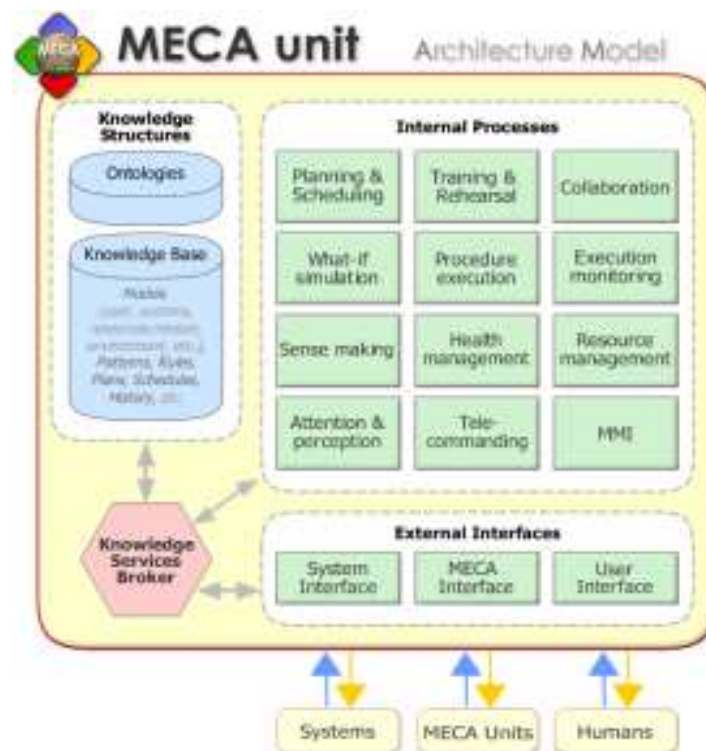


FIGURE 2. THE MECA ARCHITECTURE.

From this perspective, we can describe every *MECA unit* (as an instance of the MECA ePartner software is called) as a system capable of providing (or seeking out) certain Knowledge Services. A MECA unit should be

able to provide the basic service in a disconnected way, but aim to using the shared resources of a network of MECA units in order to distribute the tasks and share knowledge in the most efficient way.

The SOA model is based on contracts with the service consumers, which are users or other services. In the context of MECA, the consumers can be the crewmembers, systems (rovers, etc.) or other MECA Units. Within a MECA Unit, services can be consumed by other internal services, supporting flexible combinations of processes adapting to the user needs and the operational context.

In addition, other applications can use MECA services. This can be used by the manufacturers of equipment used in the mission to save effort to develop software performing similar functions, delegating to MECA certain processes and provide results that can be consumed by the consumer application. This would require providing documentation describing the services and interfaces that MECA will make available to external systems.

The MECA Requirements Baseline and the MECA Architecture were evaluated and subsequently refined in phase 2 of the project by building and evaluating a Proof-of-concept Demonstrator, using current technology, simulation, and ‘Wizard-of-Oz’ mock-up techniques. In order to differentiate the Demonstrator implementation from the actual, future MECA software product, the former is colloquially referred to as *MECA 2007*, the latter as *MECA 2017*.<sup>1</sup>

MECA 2017 cannot currently be built, because much of the necessary technology simply does not yet exist. The purpose of the Requirements Baseline is to identify what capabilities MECA 2017 needs to provide whereas the MECA Technical Specification outlines the foreseen emerging technology required.

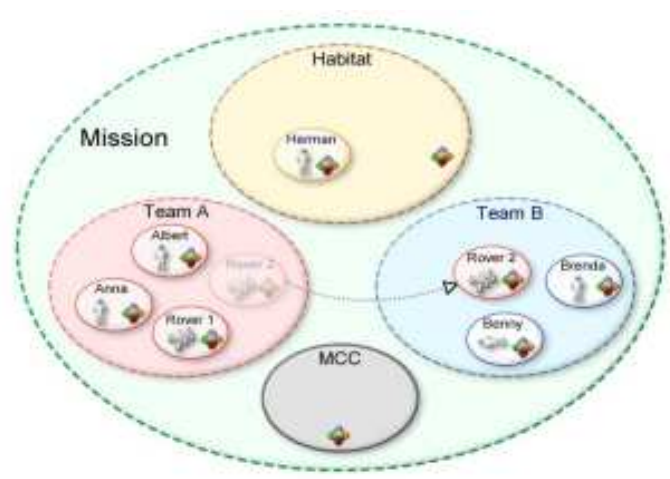


FIGURE 3. THE HYPERTHERMIC ASTRONAUT SCENARIO.

A typical example of the kind of scenario that was used in MECA is the *Hyperthermic Astronaut* scenario (see Figure 3) in which an astronaut’s spacesuit malfunctions while he is on EVA. The MECA software is expected to assist the various actors in this scenario (all in the possession of a separate, networked MECA unit) in the different aspects of handling this emergency situation. This assistance ranges from detecting the problem in the first place to rescheduling existing plans and resources so that the astronaut can be safely transported back to the habitat as soon as possible (e.g. by re-assigning a rover originally working with another EVA team) without jeopardising the safety of any of the other astronauts.

## 2.2. THE KNOWLEDGE BASE

As can be seen from the MECA Architecture described in the previous section, the different services that can comprise a MECA Unit are diverse and occupy the entire level-of-abstraction spectrum. At the higher levels of abstraction, a MECA Unit must be able to store and reason about high-level structures representing mission concepts such as goals, resources, actors (such as astronauts or other MECA Units), tasks, procedures, plans

<sup>1</sup> The ‘2017’ is a randomly chosen year — it is simply intended to signify that this is a ‘future’ MECA system.

and schedules. At the lower levels, MECA processes must have access to output from vehicles, payloads, instruments, sensors, actuators, processors, telemetry, test results, and fault diagnoses.

Additionally, in order to assist the human astronaut optimally, MECA must also be aware of human issues such as trust, task load, stress levels, and other cognitive aspects of human-computer interfacing. Following the general objective of human-machine partnership, a major part of the information and knowledge should be easily shared (i.e. understood) by both human and machine actors, in real-time.

The MECA Knowledge Base is envisioned as a distributed repository that should be capable of allowing all the different MECA Processes to store, retrieve, share, process and make sense of the data and models contained in it. Clearly, the interoperability aspect is going to be crucial: different services, systems and Knowledge Bases sources will be created by different manufacturers from various countries.

MECA envisions a Knowledge Base that is built on formal specification of data models in an Ontology language. Such a formalised data modelling lifts the semantics of the data out of the application domain, and in to the domain of separate reasoners and processing. Just as XML provides us with a common syntax format for data that can be processed by reusable components that opens the way for instance data to be automatically validated against a Schema, so does an Ontology-based approach provides a common domain modelling 'grammar' format that can in turn be used to validate instance data, generate documentation and code, and aid interoperability by the "open world" assumption of ontology-based modelling.

### 2.3. MECA ONTOLOGIES

The main goal of an ontology is to model/describe reality and provide a view on it. A secondary goal of an ontology is to provide means to represent and communicate the knowledge about a given reality. Informally, these meanings can be explained in drawings and documents, but it is more useful (and in fact crucial for a system such as MECA) to formalize as much semantic knowledge as possible, and bring it into the machine-processable world, which is precisely the kind of semantic encoding that ontologies offer.

For MECA, ontologies are of vital importance because they will eventually allow MECA 2017 systems to be built by using (semi)-automatic integration of components that with present-day technology would still need to be hand-coded or manually integrated. An example could be the automated generation of schedules using a planning module that is aware of and can act on hardware components provided by different parties. Other standardized automation goals such as verification, visualization, transformation, and perhaps even execution will also be made possible.

The following steps roughly describe the actions that need to be performed in order to create an ontology:

- Define the boundaries of the reality [domain / world] that must be described.
- Identify the entities [objects] existing in the selected context.
- Identify the characteristics [attributes] that describe each object.
- Select the categories [classes] in which entities can be grouped [classified].
- Find the relationships and interactions among entities (and among categories).

The key to this process is the capability to make abstractions and find relations.

In MECA, the following possible areas of domain modelling were identified:

#### DOMAIN ONTOLOGIES

Domain ontologies model all the concepts and entities that play a role of importance in the world in which the MECA system will exist.

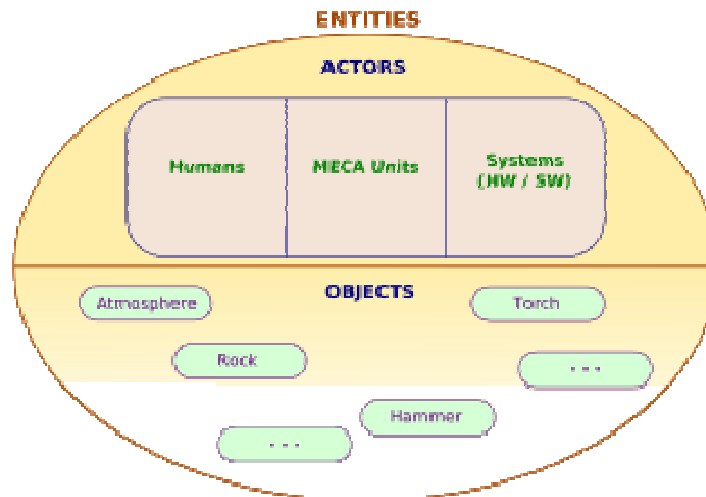


FIGURE 4. MECA DOMAIN ONTOLOGIES.

There are two major categories of domain ontologies: Object Ontologies and Actor Ontologies (see Figure 4).

*Object ontologies* model all the external aspects of the MECA universe. Relevant examples of object ontologies include for example natural environment concepts such as:

- Ecosystems (Space, planet, etc.)
- Landscape and Terrain features
- Natural resources
- Weather
- Geospatial information (Latitude, longitude, altitude, areas, etc.)
- Temporal information (Time units, periods, scales, calendars)

Also more artificial constructs such as:

- Material resources (Fuel, electricity, oxygen, water, food, waste)
- Facilities (Greenhouses, Power plants, etc.)

*Actor ontologies* are used model all concepts in the MECA universe that are capable of communication and autonomous action. Relevant examples include:

- Hardware Systems (Vehicles, Payloads, Suits, Robots, Computers, Batteries, Tools)
- Software Systems (MECA Units, Mission Control)
- Organic Systems (Astronauts, other humans, Pets, Aliens...)

Most of the Hardware Systems will themselves be 'smart' and will operate with various levels of autonomy. These systems and their functionalities can be described by their own formalisms (ontologies, XML Schema's, etc.). Mappings can be devised for connecting these systems to MECA and assuring interoperability from the conceptual level on downwards.

### CONCEPT ONTOLOGIES

Having modelled the objects and actors domains in the MECA universe, the remaining large category of ontologies is formed by the *Concept ontologies*, which model the activities, concepts, and interaction that the various objects and actors are involved with. Examples here are:

- *Tasks ontologies*: Missions, Objectives (=Goals), Experiments, Procedures, Activities (=Tasks, Actions, Operations, Steps, Behaviours), Constraints (=Conditions, Rules), Plans, Schedules, Assignments, Timelines, Timetables, Situations (Scenarios), Events, etc.
- *Communications ontologies*: Messages, Alerts, Priorities, Contexts, Channels, Interfaces, Event Log, Data archive, etc.
- *System Health ontologies*: Tests, Problems (Malfunctions, Failures, Diseases, Mistakes), Symptoms, Diagnostics, solutions (remedies, therapies), Contingency plans, Rules, etc.

- *Human Interface ontologies: cognition, task load, emotion, etc.*

A MECA scenario can be expressed as instance data of the MECA System and Domain Ontologies (*"this specific instance of class Astronaut has that many instance of class Rover available"*, etc.)

### 3. RDF/OWL EXPERIENCES & CONCLUSIONS

For the MECA 2007 Proof-of-concept Demonstrator, it was decided to choose OWL/RDF as an Ontology formalism. OWL and RDF are part of the W3C building block standards for the Semantic Web [1-5]. RDF or Resource Description Framework is used as the basic data storage format. The Ontology Web Language OWL is used to formalise the semantics and record our ontologies (for both MECA 2017 and MECA 2007).

Within the limited MECA scope, it was of course not possible to fully create all the ontologies outlined in Chapter 2, but we followed an incremental approach. We chose core functionalities in the MECA architecture that were of interest, and then modelled enough of the relevant ontology to allow implementation of that functionality in the prototype, and to allow representations of the corresponding data structures in our Knowledge Base.

The MECA Proof-of-Concept Demonstrator was implemented in Java 1.6, using Astrium-ST's *Lapap/Navigator* application framework. As RDF data store and OWL reasoner, the *Jena* library was used.

Our negative experiences with RDF/OWL can be summarised as follows:

- *Fairly steep learning curve*
- *Immaturity of tools*
- *Lack of programmatic support*
- *Modelling is not trivial*
- *Performance / bloat*
- *No chance to experiment with rules*

Our positive experiences with RDF/OWL can be summarised as follows:

- *Ontology approach successful in MECA 2007*
- *Tools are immature, but solid*
- *Rapid prototyping, data portability, separation of concerns*
- *SPARQL query language is very promising*
- *Clear potential for smooth real-time operational knowledge sharing between humans and autonomous systems on planetary missions*

We believe preparatory work for human-system knowledge sharing in a realistic mission context should be undertaken next, in order to pave the road for successful and cost-effective human-system collaboration on Moon and Mars.

### 4. REFERENCES

- [1] Frank Manola and Eric Miller (eds). RDF Primer, <http://www.w3.org/TR/rdf-primer/>
- [2] Dave Beckett (ed.). RDF/XML Syntax Specification (Revised), <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>
- [3] Patrick Hayes (ed.). RDF Semantics, <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>
- [4] Deborah L. McGuinness and Frank van Harmelen (eds). OWL Web Ontology Language Overview, <http://www.w3.org/TR/owl-features/>
- [5] Michael K. Smith, Chris Welty and Deborah L. McGuinness. OWL Web Ontology Language Guide, <http://www.w3.org/TR/owl-guide/>