Ontology to improve autonomous interactions between space vehicles

Gaëtan Séverac[1] and Eric Bensana[1]
ONERA, the French Aerospace Lab, F-31055 Toulouse, France

Future planetary robotic missions will involve several robotic agents of various origins, with various levels of autonomy which will have to interact on site, to perform tasks of increasing complexity. The different needs and problematic of such missions are presented as well as the first steps to develop an ontology covering the domain of space robotic exploration and its potential use to support direct interactions between robotic agents. This work, at the crossing of multi-agent systems, knowledge management and embedded control architectures, is aimed mainly at providing a generic service oriented interface and interactions mechanisms for robotic agents to support a high level of autonomous cooperation.

I. Introduction

The presented work is part of a prospective research program involving CNES and ONERA, related to the study of future robotic systems for planetary exploration. In the following, the term robotic agent is used as a generic term for the different kinds of device or vehicles which can be used (orbiter, rover, lander, blimp, mole ….) for space exploration. One of the objectives of the program is to prepare the transition towards missions involving several robotic agents deployed on purpose or which can be used simply because of their presence in the vicinity. Mainly because of communication constraints, it is quite impossible to consider a global and fine coordination of such a set of robotic agents from a mission control based on Earth. Consequently a major part of the coordination process has to be done directly on site by the robotic agents themselves and local peer-to-peer communications will be assumed to achieve a high level of autonomy in the way how the robotic agents interact.

The ongoing work concerns mainly the characterization of the operating capabilities of robotic agents, in order to define an *operating manual* for each robotic agent based on the notion of service. This manual will be shared with other robotic agents when a new robotic agent becomes available for the mission and updated according to events affecting the availabilities of the services it describes. In other words, such a manual will describe what a robotic agent can do and how it can do it to the attention of others robotic agents involved in the mission. The first part of this work concerns the characterization of the knowledge that a space exploration robotic agent should embed (environment, characterization of its own capacities and those of neighboring robotic agents). The second part is dedicated to the specification of high level interaction mechanisms, their implementation with respect to control software architectures embedded on the different exploration robotic agents. Finally some elements on the simulation environment which be used for the validation and testing of these concepts will be given.

II. Context of robotic planetary exploration

1. Current status

Robotic planetary exploration started in the 70's with the Soviet rovers Lunokhod which were teleoperated on the Moon and the last instance of robotic agent is the NASA Curiosity rover currently on its way to Mars. The

[1] Department of Systems Control and Flight Dynamics, contact: surname.name@onera.fr

successes of the NASA Mars rovers Spirit and Opportunity have been widely disseminated[1] and the less publicized Cassini-Huygens mission [2], composed of an exploration probe and a lander, is another significant example of successful robotic missions that has increased our knowledge of Saturn and Titan.

Actually in such mission, local interactions, between for instance Mars orbiters and rovers are limited to data storage and communications relay, and are still supervised and controlled by the ground control. Typically the activities are scheduled on Earth, before being sent for on board execution, the planning horizon depending on the capabilities of the robotic agent. To deal with non nominal situations, the strategy is to develop on-board procedures to guarantee the survival of the robotic agent, the diagnosis of the situation and recovery actions being elaborated on Earth. One step further has been made in the Cassini-Huygens mission when the Cassini probe had to interrupt the pointing of its antenna to Earth in order to follow the Huygens lander to be able to collect and store the data transmitted during the critical entry, descent and landing phase (EDL). The procedures for the monitoring of the EDL phase of the lander where fully automated and stored on board.

Efforts are made in order to increase the autonomy of robotic agents with regards to ground control. Several autonomy scales have been defined, mainly from a control and command link between the robotic agent and the ground control. The ESA Autonomy Level scale for Space Robot Systems presents a classification to describe the autonomy level in planning for the robotics missions, where the higher level E4 corresponds to "Execution of goal-oriented mission operation on-board" meaning that the robotic agent is able to autonomously interpret a goal which is sent to it (it knows how to plan and monitor the set of actions needed to achieve it).

2. Future missions and their specific constraints

Future robotic missions are designed either for the preparation of future manned missions to the Moon or Mars, the exploration of planets inaccessible to humans because of distance or too hostile environments (like Venus for instance) or even for the exploitation of local resources (e.g. for the production of propellant or water for inhabited facilities[3]). Given development constraints of space robotic agent (design, launch, cost, computing, power generation etc.), the future of space robotic exploration lies more in the development of missions involving several robotic agents coming from different agencies than in the design of costly multi purpose robotic agents.

Despite their obvious complexity, multi-vehicles exploration missions have several advantages:

- the exploration system could be incrementally built by sending successively robotic agents dedicated to specific tasks;
- existing robotic agents on site but not initially designed for a new mission can be opportunistically used;
- by sharing common resources like data acquisition, computing power, mass memory or communication, robustness of the whole system can be improved.

The context chosen for our study is restricted to planetary exploration mission without any human presence on site. The planet environment (like Mars or Titan for instance) allows the combined use of orbiters or passing-by probes in space and rovers of various sizes and aerial robotic agents (helicopters or balloons) on the surface. Current and planned missions provide plausible assumptions about the expected evolution of space robotic agents. In this context, a wide range of scenarios, like crater or canyon exploration, with different levels of complexity can be defined and thus provide experimental material for validation and tests in a simulated environment.

3. International collaborations and standardization trends

To face the development cost of these missions, the collaboration and cooperation between space agencies are becoming more and more common. Actually this cooperation is mainly at the equipment level (for instance a rover or an orbiter carrying equipments coming from other agencies). In the recently failed Phobos-Grunt mission, the Russian probe was associated with a Chinese satellite. The ExoMars mission, up to the recent NASA participation cancellation, considered two rovers (one from ESA and one from NASA). It is reasonable to think that the number of these collaborations will increase in the future, for the benefits of the scientific results and that more and more the cooperation will rely on the combination of different robotic agents which may share only partially a common mission goal.

To fulfill this type of missions, the interoperability of the robotic agents, and consequently the standardization of direct interactions between them, will be the key point. The assumption is made that in such future missions communication and knowledge representation standards will be used to allow a more efficient collaboration between robotic agents themselves. Actually, some steps have been made like for instance, at the communication level, the NASA Proximity-1 protocol [4]. which has been tested among different NASA robotic agents (rover, orbiter), but which has been tested also with an ESA orbiter used to relay NASA rover data. Another example is the "Spacecraft Onboard Interface Services" (SOIS) specification effort, made by the "Consultative Committee for Space Data Systems" [5] which describes the concepts of a service interface board. It provides an overview of the interface (services and concepts) and give some recommendations for the use of certain services and aims to standardize the interfaces between the different facilities of a spacecraft. An interesting example is the specification of "Device Discovery Service" [6] which can detect whether new devices become active and thus change the hardware configuration of a spacecraft. For ESA satellites, the Packet Utilization Standard (PUS) [7] promotes a similar approach of standard services description and allows to specify the kind of possible interactions for a given satellite, by giving the set of services available on board and the description of the protocols needed to use them.

III. Problematics

1. Objectives

The main objective is to increase the autonomy of the overall planetary exploration system by improving direct local interactions between robotic agents, as a continuation of the actual efforts to develop standards and interoperability [4], [7–9]. To support the standardization process, the proposal is to develop an ontology dedicated to space exploration. When a new robotic agent is developed, this ontology will be used to specify and implement an In Situ Interaction Service (ISIS) software component that will be integrated into the robotic agent. This component will support, once on site, local communications, exchanges and cooperation with already present robotic agents, having also an ISIS component and thus sharing the same knowledge standard. From a flight software point of view, the ISIS component will be an additional function of the control software architecture of the robotic agent, in charge of local communications and supporting high level interactions between robotic agents.

2. Toward a dynamic network of robotic agents

Globally the set of robotic agents composing the exploration system can be thought as a dynamic network where nodes are the agents and edges represent loose relations between these agents, like *IsKnownBy* or *CanCommunicateWith* etc., which only state that two robotic agents have a mutual knowledge of their capabilities and that they can interact in one way or another. This virtual network is dynamic because it has to evolve in time according to internal or external events like the arrival or the departure of robotic agents from the exploration site, the reception of ground control requests for authorizing or canceling the sharing of a robotic agent, local failure within a robotic agent or external effects (weather conditions, dust …) impacting the availability of agents, loss of communication etc. To generalize all this type of situations, the capabilities of the robotic agents will be described based on the notion of service, and the virtual network of robotic agents involved in a mission at a given time will be directly linked to the current availabilities of the different services of each agent.

Each robotic agent can be described as a set of equipments (power source, sensors and actuators) and an embedded control architecture including an ISIS component. As the ISIS component is in charge of the local communications, it directly supervises the local communication equipments (antennas, receiver, transmitter etc.). It will also manage an embedded knowledge base where information about the environment of the robotic agent (including other known robotic agents) will be stored as well as its own description. The ISIS component will be also responsible to inform its neighbors about its own service availability (for instance when a service becomes unavailable because of a hardware failure or because of preemption by a more important mission).

3. Service oriented description of robotic agents

From a cooperation perspective, each robotic agent will be seen as a provider of a set of services which can be possibly used by other agents and as a consumer of services provided by others. Several definitions of service have been proposed in the literature. A service can be seen as a unit of work done by a service provider to meet the need of a service consumer [10], and it has to be described through a contract that describes the "what" is achieved and implemented, but certainly not the "how" it is achieved [11].

The notion of services has been widely studied in the area of Service-Oriented Architectures (SOA). In SOA the accent is put on communication. The architecture of the system is there composed of three main parts: the Service Provider, the Service Requester (or Client), and the Service Registry, which facilitates service discovery by the requesters in centralized network architecture [12]. SOA contributed to the development of Web Services and present several benefits [14] such as i*nteroperability* (as long as an agent complies with the service protocol, it will be able to interact with the system), *code reuse* (a functional service in a system can be easily adapted to another system) and support in the *design, evolution* and *maintenance* of the system. CyberPhysical Systems (CPS), where software agents and physical devices are connected together, can be seen as an extension of these principles [14]. Recently, Service Oriented Robot Systems (SOMRS), where all agents of the system are robots, use also the notion of service to achieve cooperation, but like in SOA solutions they are still centralized, with the use of a service registry.

In our context, where robotic agents may or may not cooperate, centralization should not be thought as a starting architectural choice and each robotic agent should have its own partial service registry (at least those for the set of other robotic agents it knows). Nevertheless, it should be also possible that one or several robotic agents, because of their capacities, may be used as a service repository for a part of the exploration system (for instance an orbiter could play this role).

Even if running a given service can be a complex task for a robotic agent, from an external point of view, the only things which matter is to know:

- what are the available services: list of services associated with their status (available or not) or with a predicted timeline on a given horizon specifying the future time intervals of availability,
- what is the protocol to follow to implement the service between two robotic agents: services parameters, physical constraints etc.,
- what can be expected from the service execution: goal achieved, data collected, how information is exchanged (rate, mode of acquisition, format etc.).

Services and how they are used are then able to fully describe the possible use of robotic agents and can therefore be considered as an *operating manual [15]*. It is important to distinguish the description of services from their implementation and the description of the functionalities on board supporting them.

To ensure interoperability among the robotic agents in the service sharing process, all the robotic agent have to use the same representation of the associated knowledge. They must share concepts about their environment like weather condition, type and quality of land, types of structured data which can be exchanged (map, mission goals etc.). All those information could have associated descriptors to precise their quality (e.g. freshness, reliable source of information, accuracy). All these concepts and the possible relations between them will be described in an ontology.

4. Characterization of a service

As previously stated, services in SOA must be defined by at least: the goal that it can achieve, the conditions and constraints of its use and the protocol to access to it. This description can be refined to match services which can be present on a robotic agent. The description of a service should be structured in three main parts:

Service definition: what information is needed in order for a robotic agent to decide if the service is helpful to it.

Identification of the service and its provider like for instance “MonitoringZone_Blimp” for a service on board a blimp giving images of the zone it flies over;

Service type:

“Physical Service”: Which uses actuators with some kind of physical effect on the environment (movement, drilling etc…)

“Software Service”: Computing, data storage, communication (even if radio-wave emission is a

physical effect)...
Service achievement: Describe what the service is suppose to do, and the type of return you can get when it runs like for instance : to give images at a periodical rate of a portion of its environment.
Service mode: what kind of interactions is required between the producer and the consumer
"Active": The service's consumer periodically polls the service's provider to get feedback about the execution.
"Passive": The user invokes the service and expects a return from the provider when the goal is achieve.
"MixedMode": The service's user can polls the service's provider and/or wait for the the final return.
Service visibility: The service can be public or restricted to certain agents, under certain conditions and certificate authorization.

Consumer's view of the service: How to invoke and use the service, from the point of view of the consumer
Initialization and *execution* constraints: conditions required to invoke and run the service (environmental condition, timelines, agent state...).
Data-flows associated to the service:
Inputs: Description of the data consumed by the service. Data can be messages, arguments...
Outputs: Description of the data provided by the service. Data can be messages, arguments...
Side effects of the service: Describe what can possibly occurs and under which conditions.
Cases of failures: List and description of possible failures of this service. For example a failure of the service "MonitoringZone" is "UnreachableZone".

Provider's view of the service: How to execute the service, from the point of view of the provider
The generic term *resource* represent all that is needed to run the service (equipment, power, function, state, mode etc.)
Service state: current state of a service (available, unavailable, init, exec, end...)
Internal agent function
Internal agent state or mode required
…

IV. Approach to the problem

Beside Service Oriented Architecture already presented in the previous section, the ongoing work has also closed relationships with the domains of multi-agent systems, operating knowledge management and embedded control architectures of multi-robot systems. In the following, the focus is set on the knowledge representation and management, through the presentation of the ontology under development and the way it could be used to support interoperability and cooperation between several robotic agents.

1. Development of a specific ontology

1. Related works in Knowledge sharing and Ontologies

All the multi-robot applications need knowledge representation and sharing between agents to support intelligent cooperation mechanisms [16]. Numerous researches have been done on knowledge representation and reasoning such as the Knowledge Interchange Format (KIF) [17]. Applications of formal knowledge representation and semantic reasoning to multi-robot are evident to support knowledge inferring [19,20].

An approach that seems particularly interesting is to use ontologies. From the point of view of philosophy, an ontology represent a theory about the nature of being or the kinds of existent [20]. From a computing point of view ontologies are related to description logics and they are used as a common representation of a specific domain that allows different individuals to share concepts and rich relations among them [21]. The most commune way to implement ontologies actually is to use the OWL[2] language, which is a specialization of the RDF/XML language,

[2] Web Ontology Language (OWL) - http://www.w3.org/TR/owl-features

specified by the World Wide Web Consortium (W3C). As any modeling activity, each ontology reflects choices of its authors (even if it is designed with the aim to be generic) and does not guarantee the sharing of information but simply allows it [22].

In robotics, ontologies are used to specify and conceptualize a knowledge accepted by a community, using a formal description to be machine-readable, sharable [23] and to reason over that knowledge to infer additional information [24]. Some examples of the use of ontology in robotic applications are:

- The *RoboEarth* European project *[25]* which aims to represent a world wide database repository where robots can share information, about their experiences, with abstraction to their hardware specificities.
- The *Proteus project* [26] which supports scientific knowledge transfer between different robotics communities of France.
- The *A3ME ontology* [27] defines heterogeneous mobile devices, to allow communication interoperability between them, independently to the hardware platform, to the OS or to the communication system.
- The *SWAMO* NASA project [31,32] uses ontology as a prototyping method to provide standard interfaces to access different mission resources (sensors, agent capabilities...) of spacecrafts. SWAMO ontology is used both to define a commune description of the knowledge (application domain and agent-based control system for sensor web) and help designer in the conception and implementation of the system.

2. *Ontology development*

The ontology development is made with the widely used tool "Protégé" [30], which allows to encode the ontology in the OWL format. Existing ontologies structures like those of the SWAMO and A3ME projects have been used as inspiration and refined to fit our needs. The concept of service is central in our approach. So the ontology must allow to describe any kind of service possibly available in the context of robotics space exploration and should make this description precise enough to can be understandable and exploitable by other robotic agents. But it also describes all the knowledge about the robotic agent (environment, structure...).

The ontology is decomposed in two main parts:

- *A taxonomy* of the concepts which describes what are the concepts of the domain and allows to hierarchically organize them like for instance:

```
|-- Agent: That can be rover, orbiter, lander ... or human agent.
|-- AgentService: Enhance the different kind of services proposed by robotic agents.
|       |-- CommunicationService: e.g. "CommunicationRelay".
|       |-- NavigationService: e.g. "MoveTo", "ExploreZone", "LocalizeObject".
|       |-- AnalysesService: e.g. "DataAcquisition", "CollectSample".
|       |-- ComputingService: e.g. "ComputePath", "StoreMap".
|       |-- ...
|-- Data
|       |-- AuthorizationsData: e.g. Certificate or authentications.
|       |-- MessagesData: Different kinds of messages.
|       |-- PhysicalEnvironmentData
|       |    |-- Coordinates
|       |    |-- WeatherConditions
|       |    |-- ...
|       |-- ...
|-- ...
```

- *The relations between the concepts*, which represent another way of linking concepts by describing binary relations like for example a relation h*asAgentID* linking the concepts *Agent* and *AgentID* or other relations like *hasAgentMode*, *hasPosition* (linked to an instance of the *Coordinate* concepts), *hasAuthorization*... In the same manner, relations are defined that can be set between the Services concepts and others concepts like *hasServiceID*, *hasServiceInput* (linked to an instance of a needed data type to execute the service), *hasStat*e... The figure 1 shows a schematic representation of those links. In the ontology, it is possible to set up groups of relations like "AgentProperties" and "ServiceProperties".

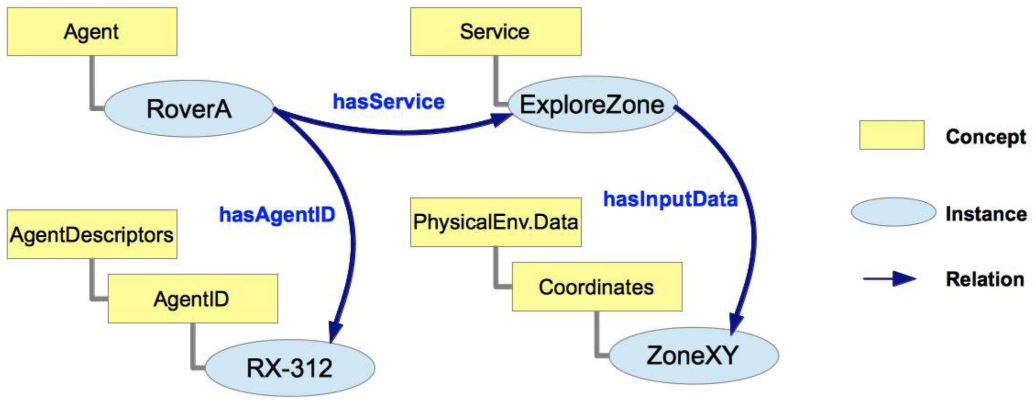

Figure 1: Schematic representation of relations linking instantiation of concepts in an ontology.

2. From ontology to ISIS implementation

The methodology proposed is organized in three steps:

- **Generic ontology development**: First of all an ontology will be created to model the environment of the space exploration missions. This ontology will be centered around the notion of service offered by the agents, and will include the general knowledge associated (physical environment, robotic agent description, services description...). This ontology kernel can be seen as one major output of the standardization process and is not restricted to a given mission but should cover as much as possible the whole domain of space exploration.

- **Specific ontology adaptation**: Then a specific instance of this ontology will be produce to cope with a particular mission, involving particular robotic agent(s), services and environment conditions. This final ontology representing a given robotic agent and its future environment, will reuse a part of the kernel and extensions or refinements induced by the characteristics of the robotic agent. For instance at this stage, interfaces with the embedded software should be defined as well as information about implementation.

- **Ontology transcription**: The last step consists in making this agent specific ontology suitable for an onboard implementation and developing the ISIS component for the robotic agent and integrating it within the software architecture of the agent. This step should use as much as possible automatic code generation to avoid errors. Typically an ontology can be dumped in an OWL format (XML like description), but there are few chances that we will used directly this format. One of the options is to build a translator to get a kind of object-oriented representation of the ontology like it can be done when converting XML files into objects and develop ad-hoc functions to deal with that representation. Another option is to use the existing RoboEarth ROS[3] package which allows to encode an ontology in Prolog and perform reasoning tasks based on this logical model. At the present moment, the final choice has not yet been made, but we will begin the experimentation with the second option for sake of simplicity.

It is important to quote that this approach respect the diversity of origins for the robotic agents. It only impose that the same ontology is used as starting point (like any standard) but does not impose a way to build, for instance, the embedded software. It requires only that such software should be able to accept an ISIS module running as another function.

[3] Robot Operating System (ROS) - www.ros.org

3. Interaction mechanisms

The development of the ontology and of the functions and models which will be present within any ISIS component should take into account how this knowledge will be used. The context of robotic space exploration lead us to consider different types of interactions depending on some mission phases. Three main cases have been identified: discovery, exploitation and updating.

1. Discovery case

This phase concerns the situation where a new robotic agent arrives in, or is made available for, an established network of robotic agents (for example after a spacecraft landing when a new rover arrive in an area where other robotic agents are operating). In this case the newcomer must provide to the others robotic agents information about its capacities and should receive and store the capacities of its neighbors. This discovery phase can be in initiated autonomously between robotic agents when they have embedded mechanisms to detect the presence of a new agent (a rover discovering another robotic agent in its surrounding), but it could also be initiated by the ground control if this mechanisms are not present or not adapted (a rover cannot discover a new orbiter by itself because of communication constraints).

When a robotic agent A discover a service S_B on robotic agent B, there are two kinds of situations:

- the service itself or all concepts associated with a new service are already known: in this case the agent is able to interpret the dependencies and the goals attached and it must create a new service instance in its knowledge base (setting also associated links with other concepts);
- at least one of the concept attached to the service is unknown: in this case the interpretation of this service is impossible by the agent and requires an update of the knowledge base of the agent (see following paragraph 3 Knowledge Updating case).

2. Exploitation case

In this case, it is assumed that an agent has in its knowledge base a representation of services provided by the other robotic agents it can cooperate with. The goal is here to exploit the knowledge of the neighbor's services description to organize collaboration between two or several agents to perform a high level objective. The panel of possible interactions is rather large. It can range from asynchronous goal to subgoals decomposition in team work planning, to a synchronized closed loop of commands in case of a robotic agent controlling another one. In these situations interaction mechanisms should be characterized mainly by the data exchange which should take place (types of data, rate, delay, size etc.). Examples of such interactions mechanisms can be:

- *Decentralized planning:* Each agent plans its own behavior and some kind of negotiation has to take place to converge to an acceptable global plan;
- *Centralized planning*: A robotic agent, with more computing power or more easily connected to the ground control, computes for the others agents in a team an action plan, and then distributes subgoals or partial plans to them;
- *Relaying Data*: A robotic agent uses other(s) agent(s) to relay a communication to its recipient; in this type of situation decentralized networking practices can be reproduced;
- *Coordinated exploration*: Specialized robotic agents collaborate on specific tasks to explore a zone of interest in order to build a map of the area and localize agents in it or points of interest (e.g. a flying agent associated to a terrestrial robot).
- ...

3. Knowledge Updating case

The last case, which is presently out of the scope of the current work, concerns the long term evolution of the fundamental concepts coded in the embedded knowledge base within an ISIS component. Given the expected life duration of some robotic agents (several years) and assuming that the standardization process on Earth goes on enriching the ontology, a new incoming robotic agent B could manipulate concepts which were unknown to a robotic agent A when it was launched.

To cope with such situations, several options can be listed. The first and simpler option is to do nothing, in this case the new agent B can use services of the agent A and agent A can use only the services of agent B it can understand. The second option consists in updating the knowledge of agent A such that both agents share the same concepts. Here again two ways of doing this updating can be considered: the first one which is the simpler is to let the ground control on Earth update the knowledge of agent A, before the arrival of agent B; the second one which would be of greater interest is to embed on B all that is needed to update A once A and B will be communicating; this solution requires that systematically an "update internal knowledge" service is present on agent A. These considerations are only preliminary statements which may impact some choices about standard services which can be implemented on all robotic agents. The feasibility of such updates, in term of memory requirements, safety, stability of the knowledge etc. would requires specific research.

V. Experimental validation

The validation of the ISIS development chain will be made by simulation. A simulation environment based on ROS will be used. In a first version, each robotic agent will be represented by two ROS nodes (one for the ISIS component and one for the rest of the control architecture – ARCH) and communications will be set between each ISIS and ARCH nodes, as presented in figure 2. Communications between ISIS nodes of different robotic agents will be established according to mission execution and the purpose of this simulation is to validate the knowledge representation and the different service-based interactions mechanisms between the robotic agents.

In future versions, the control architecture could be refined by introducing equipments (actuators, sensors) and some hierarchical structure in the control software organization.

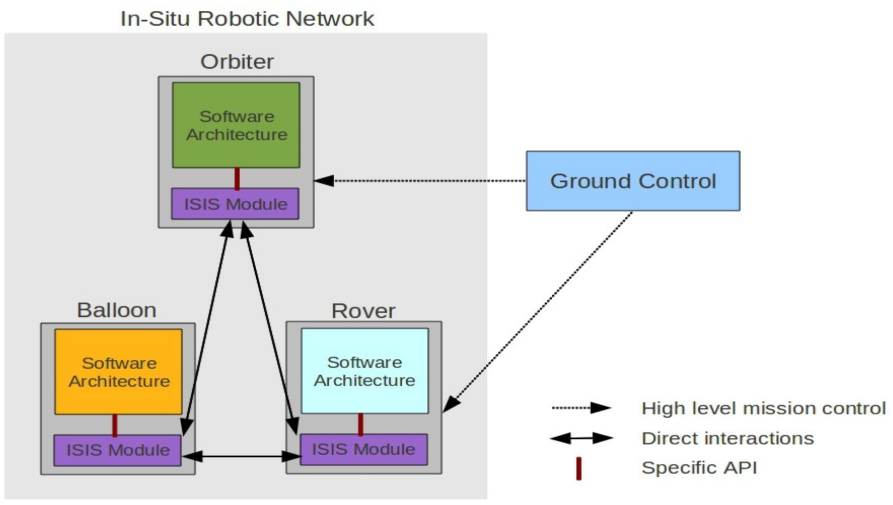

Figure 2: An architecture of the simulation system used to test the ISIS module.

VI. Conclusion

This article presents the current state of the definition of an ontology to support cooperation between robotic agents involved in planetary exploration missions. The definition of operating knowledge of robotic agents, the characterization of various interaction mechanisms between those agents and the overall implementation in embedded software architectures is made. The main objective is to increase the overall space exploration system's autonomy by allowing a higher level of local interactions between robotic agents in a context where ground control is assumed to send high level, goal oriented, commands. The proposed ontology, will be used as a tool to ensure the standardization of the knowledge modeling and thus to support interoperability. It will be used also as a specification for deriving an In Situ Interaction Service software component to embed on robotic agents who have to cooperate. The testing and validation of these concepts will be done in simulation using realistic scenarios.

References

[1] J. and others Bell, “(Nearly) Seven Years on Mars: Adventure, Adversity, and Achievements with the NASA Mars Exploration Rovers Spirit and Opportunity,” *AGU Fall Meeting Abstracts*, 2010.

[2] C. Hansen, J. Waite, and S. Bolton, “Titan in the Cassini-Huygens Extended Mission,” *Titan from Cassini-Huygens*, p. 455--477, 2010.

[3] Ashley Stroupe, A. Okon, M. Robinson, T. Huntsberger, H. Aghazarian, and E. Baumgartner, “Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance,” *Autonomous Robots*, vol. 20, no. 2, pp. 113-123, 2006.

[4] G. Kazz and E. Greenberg, “Mars Relay Operations: Application of the CCSDS Proximity-1 Space Data Link Protocol,” 2002.

[5] CCSDS, “Spacecraft Onboard Interface Services,” 2007.

[6] CCSDS, “Spacecraft Onboard Interface Services—Subnetwork Device Discovery Service,” 2009.

[7] M. Merri, B. Melton, S. Valera, and A. Parkes, “The ECSS Packet Utilization Standard and Its Support Tool,” in *SpaceOps 2002 Conference, Huston-Texas USA*, 2002, vol. 49, no. 6151, pp. 1-9.

[8] A. Tramutola and A. Martelli, “Beyond the Aurora Architecture for the new challenging applications. The Enhanced Avionics Architecture for the Exploration Missions,” 2010.

[9] S. Chien, R. Doyle, A. G. Davies, A. Jonsson, and R. Lorenz, “The Future of AI in Space,” *Intelligent Systems, IEEE*, vol. 21, no. 4, pp. 64–69, Jul. 2006.

[10] H. He, “What Is Service-Oriented Architecture,” *O’Reilly - webservices.xml.com*, pp. 1-5, 2003.

[11] R. Perrey and M. Lycett, “Service-oriented architecture,” in *Symposium on Applications and the Internet Workshops*, 2003.

[12] S. Ambroszkiewicz, W. Bartyna, M. Faderewski, and G. Terlikowski, “Multirobot system architecture: environment representation and protocols,” *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 58, no. 1, pp. 3–13, 2010.

[13] M. Papazoglou, P. Traverso, and S. Dustdar, “Service-oriented computing: a research roadmap,” in *Dagstuhl Seminar Proceedings*, 2006, no. April, pp. 1-29.

[14] J. Shi, J. Wan, H. Yan, and H. Suo, “A Survey of Cyber Physical Systems,” in *Int. Conf. onWireless Communications and Signal Processing*, 2011, pp. 1–6.

[15] D. Brugali and P. Scandurra, “Component-based robotic engineering (part i)[tutorial],” *Robotics & Automation Magazine, IEEE*, vol. 16, no. 4, pp. 84–96, 2009.

[16] J. M. Evans, E. R. Messina, and J. S. Albus, “Knowledge engineering for real time intelligent control,” in *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*, 2002, vol. 14, pp. 421–427.

[17] M. Genesereth and R. Fikes, “Knowledge interchange format-version 3.0: reference manual,” 1992.

[18] M. Compton, H. Neuhaus, K. Taylor, and K. N. Tran, “Reasoning about sensors and compositions,” in *Proceedings of the 2nd International Workshop on Semantic Sensor Networks, SSN09*, 2009, vol. 522, pp. 33-48.

[19] E. Aker, A. Erdogan, E. Erdem, and V. Patoglu, “Housekeeping with Multiple Autonomous Robots: Knowledge Representation and Automated Reasoning for a Tightly Integrated Robot Control Architecture,” in *Knowledge Representation Workshop at IROS 2011*, 2011.

[20] M. Ehrig and Y. Sure, “Ontology mapping-an integrated approach,” *The Semantic Web: Research and Applications*, pp. 76–91, 2004.

[21] K. Baclawski and A. Simeqi, “Toward Ontology-Based Component Composition,” in *10th OOPSLA Workshop Behavioral Semantics*, 2001, pp. 1–11.

[22] P. Deplanques, “Vers le test de l’autonomie des robots : une ontologie de la robotique,” Université Montpellier II, 1996.

[23] Z. Sellami, V. Camps, N. Aussenac-Gilles, and S. Rougemaille, “Ontology Co-construction with an Adaptive Multi-Agent System: Principles and Case-Study,” *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, pp. 237–248, 2011.

[24] 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*, 2005, pp. 27–34.

[25] R. Waibel, M. and Beetz, M. and Civera, J. and D’Andrea, R. and Elfring, J. and Galvez-Lopez, D. and Haussermann, K. and Janssen, R. and Montiel, J.M.M. and Perzylo, A. and Schiessle, B. and Tenorth, M. and Zweigle, O. and van de Molengraft, “RoboEarth-A World Wide Web for Robots,” *Robotics Automation Magazine, IEEE*, vol. 18, no. June, pp. 69-82, 2011.

[26] B. P. P. Martinet and B. Patin, “Proteus: A platform to organise transfer inside french robotic community,” in *3rd National Conference on Control Architectures of Robots (CAR)*, 2008.

[27] A. Herzog, D. Jacobi, and A. Buchmann, “A3ME-an Agent-Based middleware approach for mixed mode environments,” in *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2008, pp. 191–196.

[28] K. J. Witt et al., “Enabling Sensor Webs by Utilizing SWAMO for Autonomous Operations,” in *NASA Earth Science Technology Conference (ESTC2008)*, 2008.

[29] A. Underbrink, A. Potter, K. Witt, and J. Stanley, “Modeling sensor web autonomy,” *Aerospace Conference, IEEE*, 2011.

[30] “Protégé - Stanford University.” [Online]. Available: http://protege.stanford.edu.