USING ONTOLOGY TO IMPROVE AUTONOMOUS INTERACTIONS BETWEEN SPACE VEHICLES: APPLICATION TO MULTI-VEHICLES PLANETARY EXPLORATION MISSIONS

Gaëtan Séverac and Éric Bensana

ONERA, the French Aerospace Lab, F-31055 Toulouse, France. Contact: surname.name@onera.fr

ABSTRACT

In prospective planetary missions, heterogeneous vehicles such as orbiter, lander, rover, blimp will have to cooperate in-situ in order to increase the overall exploration capabilities. This paper is aimed at presenting the current status of modelling and methodological developments to support the definition and implementation of an In Situ Interaction Service for any space vehicle subject to cooperate in such missions.

The main objective of this work is to increase the overall system's autonomy by allowing local direct interactions between vehicles, with minimal intervention from the ground control. The knowledge modeling and sharing, using ontology as formalization support, as well as the implementation of associated interactions mechanisms will be the the basis to support an efficient cooperation. A simulation environment based on ROS, is currently under development to test and validate these concepts.

Key words: Multi-Robot, Space Exploration, Robot Interactions, Ontology, Simulation.

1. INTRODUCTION

The presented work is part of a prospective research program involving CNES (French Space Agency) 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 for space exploration (orbiter, rover, lander, blimp, mole...).

One of the objectives of the work 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 research 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 modelling of the knowledge that a robotic agent should embed (environment, characterization of its own capacities and those of neighbouring robotic agents) and the specification of the high level interactions mechanisms needed to share and use this knowledge autonomously. The second part propose a methodology to derive the ontological description of the agent knowledge and its implementation in the control architecture of the agents. Finally some elements will be given on the simulation environment which will be used for the validation and testing of these concepts.

2. PLANETARY ROBOTIC EXPLORATION BY MULTI-AGENT SYSTEM

2.1. Context

Actually in robotic exploration 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 which schedule all the activities.

Future robotic missions can be designed for the preparation of manned missions to the Moon or Mars, or to go on 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). But given the development constraints of space robotic agents (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 and dedicated to specific tasks – than in the design of costly multi purpose robotic agents. In such perspective, all the agents will not arrive on site in the same time, but in sequence over a time horizon which can be counted in years. This multi-agent missions will be more complex but present additional advantages if the agents are able to cooperate on site autonomously (e.g. improving mission's flexibility and reliability thanks to opportunistic team reconfiguration, computing resources sharing...).

The assumption is made that in future missions, involving different robotic agents, communication and knowledge representation standards will be used to allow a more efficient collaboration between robotic agents themselves.

For this study, the global context selected concerned the exploration of planet with an atmosphere (like Mars or Titan), performed by several robotic agents (orbiters or passing-by probes, rovers of different types and aerial robotic agents), with no human presence on site. This allows to consider a wide range of scenarios, like crater or canyon exploration, with different levels of complexity. This context is more detailed in the paper [SB12].

2.2. Problematic

The main objective is to increase the autonomy of the overall planetary exploration system by improving direct local interactions between robotic agents, keeping in mind the actual standardization and interoperability efforts. 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. Once on site, this generic component will support local communications, exchanges and cooperation with already present robotic agents, having also an ISIS component and thus sharing the same knowledge standard. The ISIS component will be an additional function of the control software architecture of the robotic agent, in charge of local communications and supporting different levels of peer to peer interactions between 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 cancelling 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, actuators, computing resources...) and an embedded control architecture including an ISIS module which will be in charge of supervising the local communications using dedicated equipments (antennas, receiver, transmitter etc.) and managing 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 module will be also responsible to broadcast its own services availabilities (for instance when a service becomes unavailable because of a hardware failure or pre-emption.

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

3. RELATED WORK

3.1. Service Oriented Architecture

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 [He03], 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 [PL03].

The notion of services in multi-agent systems 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 [ABFT10]. SOA contributed to the development of Web Services and present several benefits [PTD06] such as interoperability (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. So it is now also interesting for other software development, even in critical domain such the space operation [Sar12]. CyberPhysical Systems (CPS), where software agents and physical devices are connected together, can be seen as an extension of these principles [PTD06]. 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 be 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 to know is what are the available services and how to use them. This consists of having list of services associated with: their status (available or not) or with a predicted time-line on a given horizon specifying the future time intervals of availability, the protocol to follow to implement the service between two robotic agents (services parameters, physical constraints etc.), and the description of 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 [BS09]. It is important to distinguish the description of services from their implementation and the description of the functionalities on board supporting them.

3.2. Ontology

All the multi-robot applications need knowledge representation and sharing between agents to support intelligent cooperation mechanisms [EMA02]. Numerous researches have been done on knowledge representation and reasoning such as the Knowledge Interchange Format (KIF) [GF92]. Applications of formal knowledge representation and semantic reasoning to multi-robot are evident to support knowledge inferring [CNTT09, AEEP11].

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 [AEEP11]. 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 [BS01]. Concretely an ontology is ”simply” composed of *concepts* that can be any kind of things, linked by *relations*. The concepts can be instantiated to represent the existence of a unique entity. The most commune way to implement ontologies actually is to use the OWL language, which is a specialization of the RDF/XML language, specified by the World Wide Web Consortium (W3C)[1]. As any modelling 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 [Dep96].

In robotics, ontologies are used to specify and conceptualize a knowledge accepted by a community, using a formal description to be machine-readable, sharable [SCAGR11] and to reason over that knowledge to infer additional information [SM05]. Some examples of the use of ontology in robotic applications are: The RoboEarth European project [Wai11], which aims to represent a world wide database repository where home service robots can share information, about their experiences, with abstraction to their hardware specificities. The Proteus project [MP08] which supports scientific knowledge transfer between different robotics communities of France, by representing robots and scenarios in an ontology to guaranty a commune and standardize use. The A3ME ontology [HJB08] defines heterogeneous mobile devices for crisis situations application, to allow communication interoperability between them, independently to the hardware platform, to the operating system or to the communication system. The SWAMO NASA project [WSS$^+$08, UPWS11] 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 (satellites or rovers).

4. MODELING OF ROBOTIC AGENTS

4.1. Common knowledge representation

The purpose here is to represent all the knowledge of an agent in an unambiguous manner, ontology offer a good solution to model the representation chased for this knowledge, but previous reflections have to be done before about the concepts that should be represented and about their organization. An additional interest of having

[1]Web Ontology Language : http://www.w3.org/TR/owl-features

a knowledge representation well organized in an ontology, is that it will allows to refine it, if there is a need of adding new concepts, without modifying all the previews concepts. To easily support new missions definition, with new agents description based on the same knowledge representation.

The knowledge of an agent can be decomposed into different parts :

1. the description of the physical environment (a map, the nature of the soil, the atmosphere, the weather, time ...)
2. information related to the global mission, the robotic agent is involved in like phases, goals, constraints etc.
3. the different messages that can be exchanged with their associated communication protocol or recovery policy;
4. the description of an agent (physical description, identification, capacities...).

In the case of cooperation between robotic agents, it is also important to distinguish between the part of knowledge which is public and shareable, and the part which should remain private, only dedicated to the use of the owner agent itself, and which should not be propagated in the network. This second part may concern sensible data or restricted functionalities, but also information strongly related to the specific architecture (hardware or software) of an agent. This represents additional information that should be also contained in the knowledge modelling.

A very important part of the description of the agent is the representation of its capacities, in a service oriented vision. In other words, how to characterize the services to allow agents to autonomously understand their achievements, dependencies, restrictions and the protocol to implement to use them properly. Given the state of the art and a first analysis, the description of a service could be structured as following, decomposed in three main parts:

The **Service definition** is the required information that is needed by a robotic agent to decide if the service can be helpful to it. This part contain the *Identification* of the service and its provider like for instance ”MonitoringZoneBlimp” for a service on board a blimp giving images of the zone it flies over. The *Type*, Physical (uses actuators, with some kind of physical effect on the environment or Software (Computing, data storage, relay communication). The *Achievement*, that 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. The *Mode* defines 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 – or *Mixed mode* of both where the service's user can polls the service's provider and/or wait for the the final return. And the *Service visibility*, the service can be public or restricted to certain agents, under certain conditions and certificate authorization.

The **Consumer's view** of the service should answer to questions like how to invoke and use the service, from the point of view of the consumer. This include the *Initialization and execution constraints*, the conditions required to invoke and run the service (environmental condition, timelines, agent state...). The *Data-flows associated to the service* Inputs (Description of the data consumed by the service. Data can be messages, arguments...) and Outputs (Description of the data provided by the service). The *Side effects of the service:* which describe what can possibly occurs and under which conditions. And the *Cases of failures*, a list and description of possible failures of this service. For example a failure of the service ”MonitoringZone” can be ”UnreachableZone”.

The **Provider's view** of the service should provide information on how to execute the service, from the point of view of the provider. The generic term resource represents all that is needed to run the service (equipment, power, function, state, mode etc.). This part includes, among others, the *Service state*, current state of a service (available, unavailable, init, exec, end...) and the *Internal agent requirements*, the internal agent function, state or mode required to execute the service.

4.2. Interaction mechanisms

As presented in figure 1, two main types of interactions should be managed by the ISIS module.

Internal interactions deal with communications inside the agent, mainly with the other modules of the control architecture software like sending requests related to service execution (start, stop, resume etc.) or acquiring information about the state of the robotic agent for updating ISIS's knowledge. In the other way, the local control architecture depends on ISIS for sending requests to other robotic agents or getting information from the knowledge base like for instance getting a list of available services in the neighborhood to feed its own planning activity.

External interactions deal with communications with other robotic agents and concern the management of the knowledge related to the state of the dynamic network (arrival and departure of agents) and the state of the associated services (add/remove service, update service, update environment). It concerns also requests coming from other agents for service execution (start, stop, monitor a service) or propagation of propagating control goals within the network.

All these interactions mechanisms will be described by attributes like the communication protocols supporting it, the error management policy etc.

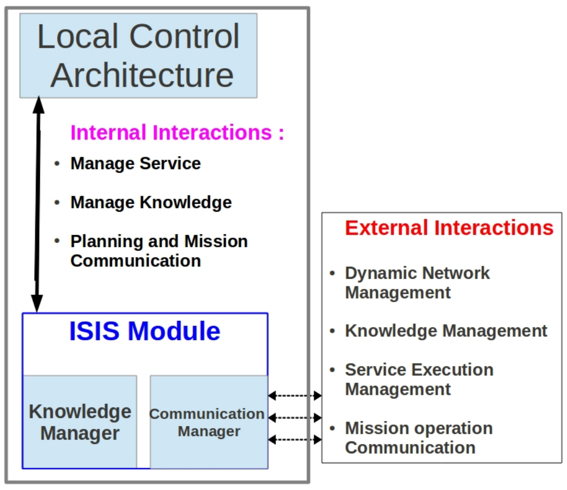
Local Control Architecture
Internal Interactions :
Manage Service
Manage Knowledge
Planning and Mission Communication
ISIS Module
Knowledge Manager
Communication Manager
External Interactions
Dynamic Network Management
Knowledge Management
Service Execution Management
Mission operation Communication

Figure 1. Simplified view of the ISIS Module interactions

5. METHODOLOGY

5.1. From the generic ontology to an ISIS module

The methodology proposed to go from the ontology knowledge representation to the implementation of an ISIS module is organized in four steps:

Generic ontology development: First of all an ontology will be created to model the global context of the space exploration missions. This ontology will be centred around the notion of robotic agents and services offered and used by the agents. It 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: When a new robotic agent has to be designed for an already ongoing or a new mission, a specific instance of this ontology will be produced to cope with the given mission (environmental conditions) and centered on the definition of this robotic agent and of the services it can provide. This instanciated ontology representing a given robotic agent in 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: This step consists in making this specific agent ontology suitable for an on-board implementation by integrating it within an ISIS component. The transcription 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 package which allows to encode an ontology in Prolog and perform reasoning tasks based on this logical model.

ISIS Core implementation: Finally, the ISIS component should be integrated within the software architecture of the agent. It will have standards functions in charge of managing the knowledge implementation, to keep it up to date and propagate it on the agent network. This module will be linked to the specific architectures of the agents using a dedicated interface to remain independent of it and of its implementation.

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, that will implement standard interfaces for interactions.

5.2. Ontology development

The ontology development is made with the widely used tool Protégé[2], which allows to encode the ontology in the OWL format. Existing ontologies structures like those of the SWAMO and A3ME projects (see section 3) have been used as inspiration and refined to fit our needs.

An ontology consists in two main parts:

A taxonomy of the concepts which describes what are the concepts of the domain and allows to hierarchically organize them.

The relations between the concepts, which represent another way of linking concepts by describing binary relations like for example a relation hasAgentID 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), hasState...

The figure 2 shows a schematic representation of those links. In the ontology, it is also possible and useful to set up groups of relations like AgentProperties and ServiceProperties.

[2]The Protégé ontology platform - http://protege.stanford.edu/

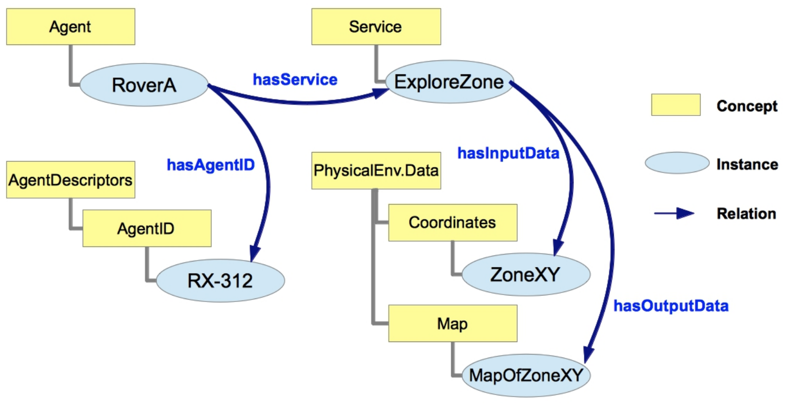

Figure 2. Schematic representation of relations linking instantiation of concepts in an ontology

5.3. Embedded code

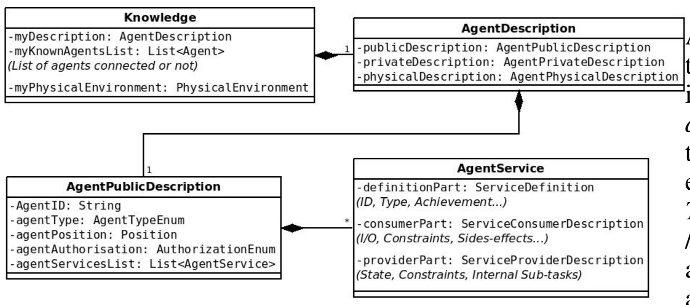
Knowledge
-myDescription: AgentDescription
-myKnownAgentsList: List<Agent>
(List of agents connected or not)
-myPhysicalEnvironment: PhysicalEnvironment
1
AgentDescription
-publicDescription: AgentPublicDescription
-privateDescription: AgentPrivateDescription
-physicalDescription: AgentPhysicalDescription
1
AgentPublicDescription
-AgentID: String
-agentType: AgentTypeEnum
-agentPosition: Position
-agentAuthorisation: AuthorizationEnum
-agentServicesList: List<AgentService>
*
AgentService
-definitionPart: ServiceDefinition
(ID, Type, Achievement...)
-consumerPart: ServiceConsumerDescription
(I/O, Constraints, Sides-effects...)
-providerPart: ServiceProviderDescription
(State, Constraints, Internal Sub-tasks)

Figure 3. UML representation of a part of the agent knowledge implementation

The ISIS module is mainly composed of a communication manager and a knowledge manager. The communication manager deals with messages sent to and received from other robotic agents. The knowledge manager keeps up to date the knowledge base, according to requests coming from the communication manager or monitoring changes in the local agent configuration (software or hardware).

As explained in the section 4, the knowledge is decomposed in several parts and figure 3 presents a part of the associated UML model. Some extracts are detailed here: In the agent description, the physical description can be seen either implemented as a functional view composed of a list of systems and subsystems describing the material and functionalities of the robotic platform and payload, either as a geometrical conception and simulation point of view, with the description of physical parts and their mechanical constraints and interaction. The map and environment meta-data (actual and forecasted weather, ground description, agents, area of interest...) could be represented using parts of the standards from Geographic Information Systems (GIS), as those proposed by the Open Geospatial Consortium (OGC) for instance.

An important part of the description of the services is the representation of the service achievement, actually it is done by specifying the kind of achievement: *UpdateKnowledge* (variable update, environment modification...), *ProvideData* (send back a data that will not necessarily be used to update a commune knowledge) or *TransmitMessage*. Then the relations between the inputs / outputs data of the services and the general knowledge are describe in an ad-hoc manner, for example for the achievement of a service of exploration, it will be state that the knowledge of the map in the zone to explore, given in input of the service, will be updated: "*Knowledge.Map(Service.Input:Zone)*". This achievement description could also have been done using predicates as in planning activities, but the use of predicates to describe the service should be as minimal as possible, because to understand a new service an agent will then have to also know the predicates employed, and not only the general environment knowledge. The services side-effects descriptions are based on the same principles.

6. EXPERIMENTAL EVALUATION

The validation of the proposed concepts for the ISIS development will be made by simulation. This tool will be useful to improve and validate the proposed approach, using realistic scenarios. In a first step, the modelling of hardware and details of the implementation of various functions will be limited to the minimum necessary to obtain realistic simulations.

In a second step more complex situations will be considered, like simulating environmental constraints (soil, communication etc.).

6.1. Testing Scenarios

From known missions, described in the literature, and plausible assumptions about expected evolution of space vehicles, it is possible to define scenarios involving several vehicles. The global context will be the exploration of a planet where rovers of various sizes and aerial vehicles (VTOL, blimp) will evolve. This context allows us to consider a wide range of scenarios.

The first and the simplest will be the insertion of a new vehicle, with new capabilities, in an existing vehicles network (following a spacecraft landing and the arrival of a new rover in an area).

More complex scenarios will be built around the exploration of a rugged terrain requiring the coordinated use of different vehicles, like a rover carrying a micro-rover scout and an blimp. Managing such a phase of exploration needs to implement a control loop within the rover-blimp-microRover team, where the images produced by the blimp will be used by the rover to localize the micro-rover and generate moving commands to it.

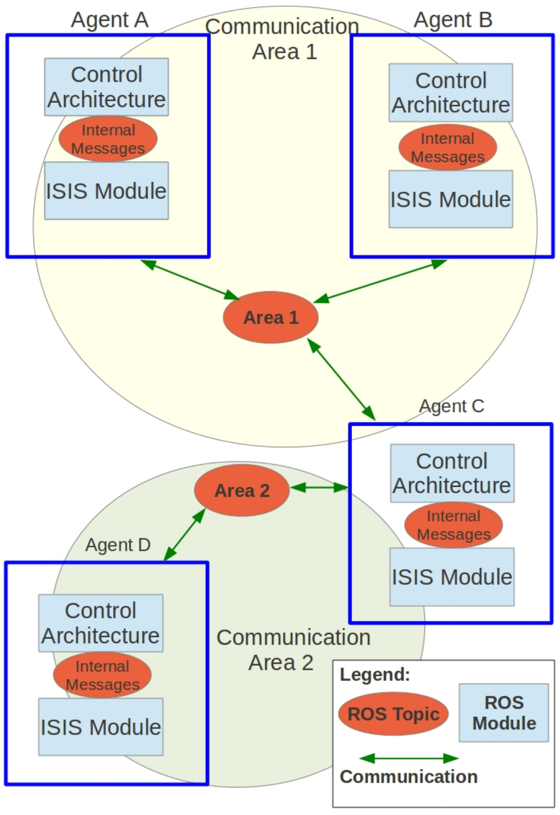

Figure 4. An architecture of the simulation system used to test the ISIS module

6.2. Implementation with ROS

A simulation environment based on the Robot Operating System (ROS) [3] 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) and communications will be set between each ISIS and control architecture nodes, as presented in figure 4.

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, the associated management functions and the different service-based interactions mechanisms between the robotic agents.

In future versions, the control architecture could be refined by introducing equipments models (actuators, sensors) and some hierarchical structure in the control software organization. If needed, existing planners, like HTN [NAI$^+$03] or the NASA EUROPA planner [FJM00] will be used during experimentation.

7. CONCLUSION

The goal of the presented work is to increase the overall autonomy of a system composed of several heterogeneous robotic agents involved in a space exploration mission where high level and goal oriented control of the whole system is required. The concepts and methodology described here are generic enough to be applied to other kind of robotic system where a strong autonomy is required (search and rescue robots, satellites network...). A service oriented model of the robotic agents, described in an ontology, have been presented. The ontology, is used as a tool to ensure the standardization of the knowledge modelling and thus supports interoperability. Then a methodogy associated to a development chain will allows to derive from this model an embedded software module (ISIS), in charge of direct interactions between agents. This module can then be implemented in the control architecture of the robotic agents. A set of associated interactions models have been studied, to determine the kind of the messages needed to allow the autonomous interactions and knowledge sharing. The next steps will be to implement the proposed solution in a simulation environment, for concepts validation and improvements.

REFERENCES

[ABFT10] 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*, 58(1):3–13, 2010.

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

[AEEP11] Erdi Aker, Ahmetcan Erdogan, Esra Erdem, and Volkan 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.

[BS01] Kenneth Baclawski and Artan Simeqi. Toward Ontology-Based Component Composition. In *10th OOPSLA Workshop Behavioral Semantics (OOPSLA 2001*, pages 1–11. Citeseer, 2001.

[BS09] D. Brugali and P. Scandurra. Component-based robotic engineering (part i)[tutorial]. *Robotics & Automation Magazine, IEEE*, 16(4):84–96, 2009.

[CNTT09] Michael Compton, Holger Neuhaus, Kerry Taylor, and K.N. Tran. Reasoning about sensors and compositions. In *Proceedings of the 2nd International Workshop on Semantic Sensor Networks, SSN09*, volume 522, pages 33–48, 2009.

[Dep96] Pascal Deplanques. *Vers le test de l'autonomie des robots : une ontologie de la robotique*. PhD thesis, Université Montpellier II, 1996.

[EMA02] 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*, volume 14, pages 421–427. IEEE, 2002.

[FJM00] Jeremy Frank, A. Jónsson, and Paul Morris. On reformulating planning as dynamic constraint satisfaction. *Abstraction, Reformulation, and Approximation*, pages 271–280, 2000.

[GF92] MR Genesereth and RE Fikes. *Knowledge interchange format-version 3.0: reference manual*. Number January. 1992.

[He03] Hao He. What Is Service-Oriented Architecture. *O'Reilly - webservices.xml.com*, pages 1–5, 2003.

[HJB08] Arthur Herzog, Daniel Jacobi, and Alejandro Buchmann. A3ME-an Agent-Based middleware approach for mixed mode environments. In *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 191–196. IEEE, September 2008.

[MP08] B.P.P. Martinet and Bruno Patin. Proteus: A platform to organise transfer inside french robotic community. In *3rd National Conference on Control Architectures of Robots (CAR)*, 2008.

[NAI$^+$03] Dana Nau, T.C. Au, Okhtay Ilghami, Ugur Kuter, J.W. Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20(1):379–404, 2003.

[PL03] Randall Perrey and Mark Lycett. Service-oriented architecture. In *Symposium on Applications and the Internet Workshops*, 2003.

[PTD06] MP Papazoglou, Paolo Traverso, and Schahram Dustdar. Service-oriented computing: a research roadmap. In *Dagstuhl Seminar Proceedings*, number April, pages 1–29, 2006.

[Sar12] Mehran Sarkarati. How to Do SOA Right? An SOA Governance Framework For The ESA Space Situational Awareness Preparatory Programme. In *SpaceOps*, 2012.

[SB12] Gaëtan Séverac and Eric Bensana. Ontology to improve autonomous interactions between space vehicles. In *SpaceOps*, 2012.

[SCAGR11] Zied 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*, pages 237–248, 2011.

[SM05] Craig Schlenoff and Elena Messina. A robot ontology for urban search and rescue. In *Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems*, pages 27–34, New York, New York, USA, 2005. ACM.

[UPWS11] Al Underbrink, Andrew Potter, K Witt, and Jason Stanley. Modeling sensor web autonomy. *Aerospace Conference, 2011 IEEE*, 2011.

[Wai11] 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*, 18(June):69–82, 2011.

[WSS$^+$08] Kenneth J Witt, Jason Stanley, David Smithbauer, Dan Mandl, Vuong Ly, Al Underbrink, and Mike Metheny. Enabling Sensor Webs by Utilizing SWAMO for Autonomous Operations. In *NASA Earth Science Technology Conference (ESTC2008)*, 2008.