An Evolvable Multi-Agent Approach to Space Operations Engineering

Sanda Mandutianu, Adrian Stoica

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, MS: 161-260
Pasadena CA 91109-8099, USA
tel: (818)354-3839, Fax: (818)393-4643

Abstract. A complex system of spacecraft and ground tracking stations, as well as a constellation of satellites or spacecraft, has to be able to reliably withstand sudden environment changes, resource fluctuations, dynamic resource configuration, limited communication bandwidth, etc., while maintaining the consistency of the system as a whole. It is not known in advance when a change in the environment might occur or when a particular exchange will happen. A higher degree of sophistication for the communication mechanisms between different parts of the system is required. The actual behavior has to be determined while the system is performing and the course of action can be decided at the individual level. Under such circumstances, the solution will highly benefit from increased on-board and on the ground adaptability and autonomy. An evolvable architecture based on intelligent agents that communicate and cooperate with each other can offer advantages in this direction. This paper presents an architecture of an evolvable agent-based system (software and software/hardware hybrids) as well as some plans for further implementation.

1 Introduction

In a complex system of satellites and ground tracking system, as well as in a constellation of satellites or spacecraft, there are some generic problems that have to be addressed. The uncertainties due to the changing environment, the necessity to react promptly to any critical event, the dynamic nature of the resource configuration, require a flexible and robust approach.

The problem of autonomy for space applications has been addressed by several agent-based contributions, such as the Remote Agent Experiment for automatically controlling and commanding the NASA DS1 spacecraft [1], or multi-agent mission support operations [12]. A multi-agent approach for mission operations has been also proposed in [8].

Agent-based architectures have been proposed for different applications such as air traffic control [10], process control [9], distributed problem solving [7], etc.

In this paper we address the synergism of the multi-agent model and evolutionary techniques in the context of a complex space and ground communication system. The evolutionary features of our agents represent the system's adaptive

nature and are meant to increase its operational robustness. Evolutionary algorithms are considered an effective way of generating or altering agent's behavior to account for changes in the physical environment. The evolutionary methods have been the object of several JPL research projects [14], [4]. The combination of evolutionary techniques and agent modeling has been a novel direction in the domain of space systems.

For past space missions, many of the sub-systems such as the attitude control, navigation, telecommunication management have been considered and operated mostly independently. As new, more complex and demanding missions are considered, the different disciplines become more interdependent. Such examples are flight through a planetary atmosphere, such as landing or ascending from another planet or the Earth, and spacecraft formation flying where the relative configuration and operation must be closely controlled.

Although in an integrated system one might equally consider the possibility of using uniform models, representations and tools, the problem can be naturally modeled as a set of autonomous problem solvers, or agents, with their own resources and expertise. In order to solve a problem requiring all their combined distributed capabilities, they have to *interact* and share common knowledge, while simultaneously maintaining their individuality and system consistency.

2 The Agent Architecture

In this paper an agent is considered as an entity described in terms of common sense modalities such as beliefs, or intentions. The agent structure we use has been conceptually inspired by Shoham's agent-based programming approach [13], with some additions from the BDI (Beliefs, Desires, Intentions) model [10], although the practical realization might differ in some aspects.

An agent-based framework includes the agent state, and an interpreted programming language used to define agents(Fig.1). The agent state is defined by four basic components:

- *beliefs*: what the agent knows about the environment and about other agents;
- *capabilities*: tasks to be accomplished by the agent under given circumstances;
- *commitments*: tasks the agent is committed to achieve (usually communicated to another agent) at a particular time;
- *intentions*: the decisions about how to act in order to fulfill its commitments .

The agent execution cycle consists of the following steps: processing new messages, determining which rules are applicable to the current situation, executing

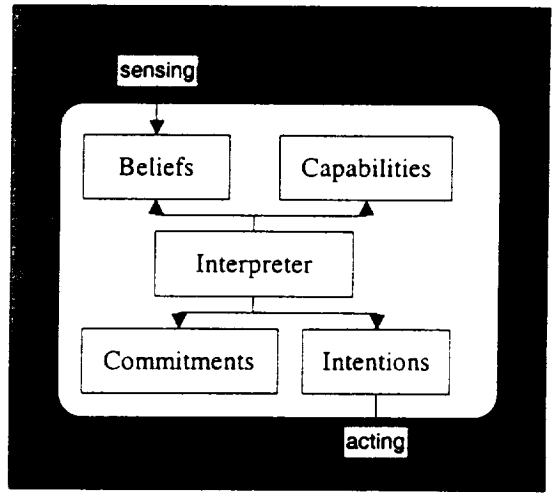

Fig. 1 Agent Structure

the actions specified by these rules, updating the mental model in accordance with these rules, and planning new actions.

The agents use KQML (Knowledge Query and Manipulation Language) [3]. KQML as both a language and a protocol can be viewed as being comprised of three layers: a content layer, a message layer and a communication layer. The content layer is the actual content of the message, in a particular representation language. KQML can carry any representation language,

including logic languages, ASCII strings, etc.

A KQML message conceptually consists of a performative, associated arguments including the real content of the message and a set of optional arguments describing the content in a manner that is independent of the content language. For example, a query about the availability of a particular antenna might be represented as:

```
(ask-all :content (AVAILABLE (?antenna))
         :ontology MISSION-MODEL)
```

The applications are considered as knowledge based agents sharing their knowledge by using an agent communication language- KQML. The infrastructure for such an open knowledge sharing system can contain different knowledge representation structures, intercommunication format(s), one or more knowledge manipulation languages, a common shared model or set of models (ontology) and a software framework to allow for the development of actual software systems[11].

4 Mission Operations Agents

To demonstrate the suitability of agent-based architecture for mission operations, we started with a simple but representative scenario where concepts such as model sharing, ontologies and interoperability can be adequately represented.

A telecommunication and radio navigation network, the Deep Space Network (DSN) supports the communication between the spacecraft and the ground stations. DSN receives telemetry signals from the spacecraft, sends commands to control spacecraft operations, generates the radio navigation data used to locate and guide the spacecraft, etc. The telecommunication link between a DSN node and the spacecraft provides navigation data as well as various other communication data.

Typically, the planning and scheduling have been coordinated for both partners- DSN and spacecraft, on the ground. Generalizing, any partner might have the ability to request data. The partner must be notified and must cooperate to configure the link properly for the requested data flow. The scheduling process has to take into account constraints and requirements that are not the same as those for the telecommunication processes only. There will be probably constraints of this sort associated with the spacecraft-to-spacecraft links or other non-Earth links.

The following agents have been identified as typical for the given scenario (Fig. 2):

- **Navigation Agent**: ensures that the spacecraft maintains its trajectory, controls and monitors other on-board activities such as capturing images, achieving science experiments, etc.
- **Telecom Agent**: controls the on-board telecommunication processes between the spacecraft and the ground stations.

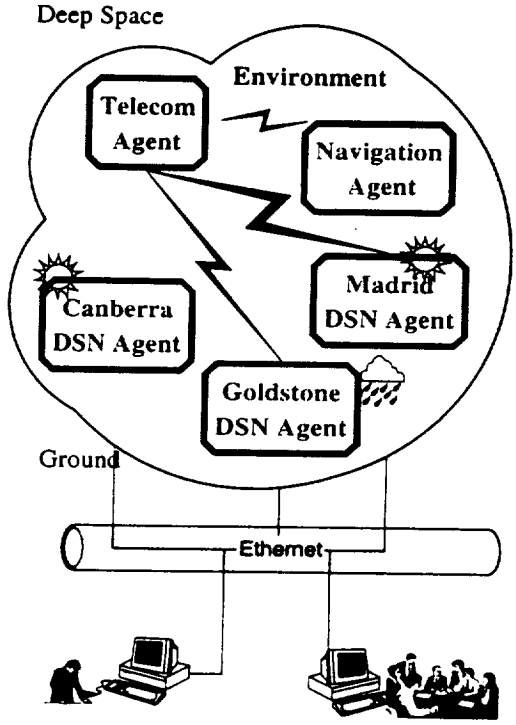

Fig. 2 Multiagent Architecture

- **Ground Complex System Agents**: control the telecommunication processes on the ground. There are as many agents as complex systems on the ground.

Some possible elements in a navigation/telecom ontology include: antenna, ground complex, navigation, and telecom link (Fig. 3).

The agents can reason about what to do

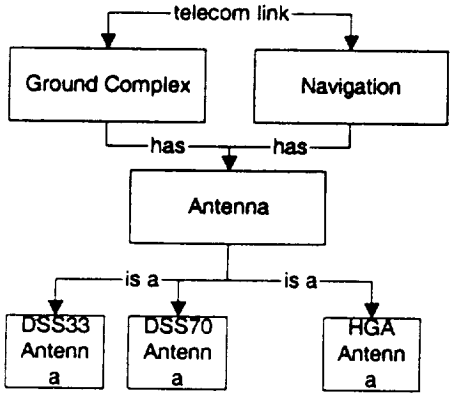

Fig.3 A semantic network for a navigation/telecom ontology

based on their current state represented by beliefs, capabilities, existing commitments, etc. Beliefs represent the current state of the agent. Some examples of beliefs, given in pseudo-code as a pair of belief name and belief value type, are given in Fig.4.

Position	(Coord1, Coord2)
Goal	(String, StartTime, EndTime, Duration, Status)
Velocity	(Float)
StartTime	(Time)
EndTime	(Time)
Duration	(Integer)
Coord1	(Float)

Fig. 4 Beliefs

The agent capabilities define the actions that the agent can perform provided their preconditions are satisfied. Capabilities, as opposed to beliefs, are given for the life of an agent. Beliefs can change, by adding new beliefs or updating existing ones. The actions can either target the environment in some way, or may accomplish a communicative act.

An agent can perform physical actions such as pointing the spacecraft to a target, turning the camera on, achieve a science experiment, send data to the ground (downlink), etc. It also can send a message to other agents asking for information, receive messages requesting goal achievement, and so on.

For example, consider the Telecom Agent has to downlink data. Some of the actions of this agent are: analyze the telecom link, choose a ground complex to downlink, and downlink. If the ground complex accepts the request to downlink, it agrees to perform the requested action at the requested time, based on the details of the request, its behavioral rules and its current mental model- beliefs, existing commitments, etc. (Fig. 5).

In general, successful execution of

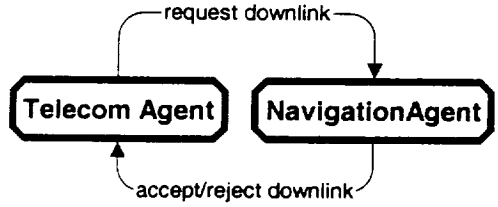

Fig.5 Inter-agent Communication

an action may be beyond agent's control. For example, a Ground Complex Agent has committed to accept to downlink on behalf of a Telecom Agent. Even if the necessary preconditions are met and Telecom Agent is able to initiate execution, the action may still fail (e.g. a crash during the transmission). The Ground Complex Agent must monitor the execution so it will be able to send a message back to the Telecom Agent to report the success or failure of the commitment.

The agents might be running on different platforms and might be using different content languages for communication.

The agent behavior is described by its behavioral rules. The behavioral rule extends the commitment rule by determining the course of action that the agent takes throughout execution.

```
("Forwarding downlink selection"

WHEN
IF
(BIND [VAR Goal <?g>])
(OBJ [INST Goal <?g>.name] EQUALS
     [VAL String "image of Europa on
ground"])
(OBJ [INST Goal <?g>.status] EQUALS [VAL
String "accepted"])
(OBJ [VAR Agent <?agent>.agentInfo.name]
EQUALS
     [VAL String "Telecom"])
THEN
(DO SendKqmlMessage ([NEW KqmlMessage],
               [VAR
Agent<?agent>.agentInfo.name],
               [VAL String "achieve"],
               [VAR Goal <?g>],
               [],
               [], [], [], [], []))
)
```

Fig. 6 Behavioral rule- forwarding a goal

An example of a behavioral rule is given in Fig. 6. The rule is written using RADL (Reticular Agent Description Language) [11]. RADL has extended the idea of a commitment rule to a general behavioral rule. Behavioral rules can be seen as production rules where the IF part matches against beliefs, commitments and intentions, and the WHEN portion matches against messages. The THEN portion represents agent's actions and belief changes performed in response to the current event, internal beliefs and external environment.

The rule in Fig.6 states that if the goal "image of Europa on ground" exists as a belief, and it was accepted as feasible (goal status is "accepted"), then it will be forwarded to the telecom agent within an "achieve" message.

The goals are opportunistically accepted, based on current beliefs and commitments. The decision making process is based on knowledge based reasoning. The plan to achieve the goals- the actual sequence of actions can be pre-defined, are given recipes, or a domain dependant planning process can generate or adapt existing plans. Goal failures result in selection of some other plan or explanations of reasons. Goals are communicated between agents as content in KQML messages in a simple declarative language understood by the partners.

5 Evolving Agent Rules

We extend the BDI model by allowing the behavioral rules to change during the agent lifetime by adding an evolutionary component to the agent control structure. Since no single set of rules can foresee all situations, the agent also starts with a given set of rules, and might evolve other, based on the interactions with the environment and with other agents. The population of problem solvers can adapt to a changing environment.

Due to their high potential for optimal solutions, evolutionary techniques (such as genetic programming [6]) have been considered the appropriate method to infer and generate new rules. Upon initiation, only the basic agent functions are implemented; it is assumed that these are realized by analyzing and embedding the necessary domain knowledge before execution (mission). Problem solving requests continuously arrive at the initial agents. The agent's local decisions may affect the system's behavior as a whole, and the decisions are triggered by changes in the environment.

An example of reorganization might be illustrated by the case when the Telecom Agent is systematically overwhelmed by requests to transmit data and is constantly behind the schedule. A solution is to first evolve another algorithm for data compression that will better perform, see for example the work on evolvable hardware for compression described in [4]. Secondly, a

new rule can be defined, with the new strategy. If we consider the possibility of having certain directly hardware implemented, finally this agent will be possibly entirely implemented as a new hardware component of the system, for best performance.

In essence, the behavioral rules represent a generic production rule:

IF x is X1 AND y is Y1 THEN z is Z1

where x is one class in the set of possible classes x in {X1...Xn}, y in {Y1...YM} and z in {Z1...Zp}.

The number of possible rules increases exponentially, considering k variables in the antecedent each with N classes, the number of possible rules is N^k.

6 Evolvable Agents in Hybrid Hardware/Software Implementation

The search space of possible rules in which agents can evolve is potentially very big. While evolutionary techniques proved efficient in coping with big spaces, they are also computationally expensive. Hundreds of generations, each with hundreds to thousands "trial" rule-bases may need to be evaluated. This perspective creates the need for very rapid rule-evaluation, which although may not be needed by the system at its regular performance stage, may be required at the evolutionary adaptation stage.

The solution we propose here is that of a hybrid software/hardware implementation., more precisely, the use of a programmable hardware component that implements the rule system, providing very rapid rule evaluation.

Several rule-based systems chips have been proposed as hardware accelerators, in particular for fuzzy rule-based systems. One particular implementation of such a processor implemented in analog VLSI is proposed in [2]. The rule-based processor is programmable and can process approximately one million rules per second.

In this context, evolution of the rule-base takes place largely in hardware, by selectively modifying the control bitstrings that program the dedicated hardware. Evolutionary experiments with hardware in the loop are described for example in [15].

The system would start using some pre-defined values (randomly generated, or, when available pre-shaped from experience) for X1 and for Y1. While the results are satisfactory, there is no need to use other rules. If the system doesn't perform well in time, some other values may be better.

7 Final Remarks and Future Work

Initial experiments, work towards hardware implementation of rule-based agents and plans for on-chip evolution has been described. The two main goals of this effort have been the integration of applications into an agent based framework and investigating agents' evolving behaviors. In the end, this will allow the evolutionary algorithms developed in JPL to be applied to evolve hardware components.

There have been several agent-based approaches to system control and in particular to spacecraft control or mission operations. We can cite the Remote Agent Experiment for automatically spacecraft controlling and commanding NASA DS1 spacecraft [1], or multi-agent approaches such as multi-operation support operations [11], and handling malfunctions in the Reaction Control System (RCS) of NASA's space shuttle [5]. Some of them solve the problem using one single agent, such as for DS1, or in the case of a multi-agent approach have no explicit model for inter-agent communication or application integration. There is some other multi-agent approaches with more elaborate models for cooperation between agents such as in [16].

Further work should focus on defining mission ontology, refining the communicative models to allow for complex interactions such as negotiation. We also plan to use the chips to evolve fuzzy rule based systems in intrinsic EHW mode (i.e.

directly on the chip). Ultimately, such chips could be part of hybrid SW/HW evolvable agents that can control the spacecraft communication.

ACKNOWLEDGEMENTS

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, and it was sponsored by the JPL Director's Research and Development Fund and by the JPL Deep Space Network Technology Development Plan, under a contract with NASA

References

[1] Bernard, D.E. et al.: Design of the Remote Agent Experiment for Spacecraft Autonomy. In Proceedings of IEEE Aerospace Conference, Aspen, 1998.

[2] Daud, T. Stoica, A. Thomas, T. Li, W. and Fabunmi, J. (1999) ELIPS: toward a sensor fusion processor on a chip. In B. V. Dasarathy, (Ed.) Proc. of SPIE Vol 3719, Sensor Fusion: Architectures, Algorithms, and Applications III, Orlando, FL, 7-9 April, 1999, pp 209-219.

[3] Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McGuire, J., McKay, D., Shapiro, S., Pelavin, R., Beck, C.: Specification of the KQML Agent Communication Language (Official Document of the DARPA Knowledge Sharing Initiative's External Interfaces Working Group), Technical Report 92-04, Enterprise Integration Technologies, Inc., Menlo Park, California, 1992.

[4] Fukunaga, A, Hayworth, K, Stoica, A. Evolvable Hardware for Spacecraft Autonomy. In Proc. of IEEE Aerospace Conference, Aspen , 1998.

[5] Ingrand, F.F., Georgeff, M.P., Rao, A.: An Architecture for Real-Time Reasoning and System Control, http://www.laas.fr/~felix/publis/ieee-exp92/ieee-diagl2h.html

[6] Koza, J. R. : Genetic Programming: on the Programming of Computers by Means of Natural Selection MA:MIT Press Cambridge 1992

[7] Liu, J., Sycara, K.. Emergent Constraint Satisfaction through Multi-Agent Coordinated Interaction, Proc. of the 5th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, Neuchatel, Switzerland, 1993.

[8] Mandutianu S., Cooperative Intelligent Agents for Mission Support . In Proceedings of IEEE Aerospace Conference, March 1999.

[9] Muller, J.P.: The Design of Intelligent Agents- A Layered Approach. IN Lecture Notes in Artificial Intelligence, Springer Verlag, 1996.

[10] Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, June 1995.

[11] Reticular Systems, Inc. AgentBuilder- An Integrated Toolkit for Constructing Intelligent Software Agents, 1998.

[12] Siewert, S.: A Distributed Operations Automation Testbed to Evaluate System Support for Autonomy and Operator Interaction Protocols. In Proc. of 4th International Symposium on Space Mission Operations and Ground Data Systems, ESA, Forum der Technik, Munich, Germany, September 1996.

[13] Shoham, Y.: CSLI Agent-oriented Programming Project: Applying software agents to software communication, integration and HCI (CSLI home page), Stanford University, Center for the Study of Language and Information, 1995.

[14] Stoica, A. Fukunaga, A, Hayworth, K, Salazar-Lazaro, C. Evolvable Hardware for Space Applications. In Proceedings of the Second International Conference on Evolvable Systems, Lausanne, Switzerland, 1988.

[15] Stoica, A. (1999) Towards Evolvable Hardware Chips: Experiments with a Programmable Transistor Array. Proceedings of the 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-inspired Systems, Microneuro'99, Granada, Spain, April 7-9, 1999

[16] Tambe, M., Johnson, W.,L., Jones, R.,M., Ross, F., Laird, J., E., Rosenbloom, P.,S., Schwamb, K.: Intelligent Agents for Interactive Simulation Environments, AI Magazine, Spring1995.