Advantages of Brahms for Specifying and Implementing a Multiagent Human-Robotic Exploration System

William J. Clancey[1], Maarten Sierhuis, Charis Kaskiris, and Ron van Hoof

NASA/Ames Research Center
Computational Science Division, MS 269-3
Moffett Field, California 943055
bclancey@mail.arc.nasa.gov

Abstract

We have developed a model-based, distributed architecture that integrates diverse components in a system designed for lunar and planetary surface operations: an astronaut's space suit, cameras, all-terrain vehicles, robotic assistant, crew in a local habitat, and mission support team. Software processes ("agents") implemented in the Brahms language, run on multiple, mobile platforms. These "mobile agents" interpret and transform available data to help people and robotic systems coordinate their actions to make operations more safe and efficient. The Brahms-based mobile agent architecture (MAA) uses a novel combination of agent types so the software agents may understand and facilitate communications between people and between system components. A state-of-the-art spoken dialogue interface is integrated with Brahms models, supporting a speech-driven field observation record and rover command system. An important aspect of the methodology involves first simulating the entire system in Brahms, then configuring the agents into a runtime system Thus, Brahms provides a language, engine, and system builder's toolkit for specifying and implementing multiagent systems.

Background

Multiagent systems were a natural outgrowth of knowledge-based systems of the 1970s, the idea of multiple, distributed sources of information and model-based processing ("distributed AI") developed in the 1980s, and the affordable, networked computing platforms of the 1990s. However, just as it has become practical to construct interacting systems of hardware and software, such as robotic assistants, GPS devices, biosensors, cameras, and the like, system builders need tools to help specify how these components are to interact in complex situations, means to test the designed processes, and an implementation architecture that is robust, modular, and amenable to runtime modifications (e.g., allowing components to leave or enter the system). We need a principled methodology for building multiagent systems (Alonso 2002).

This paper describes how the Brahms simulation system (Clancey et al. 1998; Sierhuis 2001) has been adapted to provide both a tool for specifying multiagent systems and an implementation architecture for runtime agent interactions on mobile platforms. (We call these "mobile agents," to be contrasted with the more dominant meaning of agents migrating between hosts [Vulkan 2002]). We begin with a description of the work scenario, describe how Brahms is used to model and control system interactions, and then describe two field tests in which the system is gradually developed to fit the target. Finally, we conclude with a summary of advantages and limits of the Brahms architecture for multiagent applications.

This project is a collaboration across NASA centers and other organizations; all of these people might have been listed as co-authors of this paper:

- Brahms Project Group (NASA-Ames: W.J. Clancey, Principal Investigator; M. Sierhuis, Project Manager; R. van Hoof, lead programmer; C. Kaskiris, modeler)
- RIALIST Voice Commanding Group (RIACS: John Dowding, Jim Hieronymus)
- MEX Vehicle & Wireless Communications Group (Ames: Rick Alena, John Ossenfort, Charles Lee)
- EVA Robotic Assistant Group (NASA-JSC: David Kortenkamp, Kim Shillcutt, Rob Hirsh, Jeff Graham, Rob Burridge)
- Space Suit Biovest (Stanford: Sekou Crawford, in collaboration with Joseph Kosmo, JSC).

Scenario

Our application involves exploration of Mars, in which a crew of six people are living in a habitat for many months. One long-term objective is to automate the role of CapCom in Apollo, in which a person on Earth (in Houston) monitored and managed the navigation, schedule, and data collection during lunar traverses (Clancey, in press). Because of the communication time delay this function cannot be performed from Earth during Mars exploration, and other crew members will often be too busy with maintenance, scientific analysis, or reporting to attend to every second of a four to seven hour Extra-Vehicular Activity (EVA). In the initial scenario implemented in 2002 (Figure 1), a EVA crew member drives an ATV,

[1] Also Institute for Human and Machine Cognition, UWF, Pensacola, FL.

while the EVA Robotic Assistant (ERA; Burridge & Graham 2001; Shillcutt et al. 2002) follows along or is directed to carry out certain activities. Brahms software, represented by the ellipses, is running on board the ERA and the ATV. A wireless data network (MEX/KaoS) links all components (biosensors, ERA, and Brahms running on crew backpacks, the ERA, ATVs, and in the hab). Through voice commanding (Dialog system), the crew member may control the ERA, name places, and log science data. Brahms agents carry out the commands.

Brahms Architecture

Brahms is a multiagent system, developed for over a decade as a way to model and simulate people within a social and physical world. Agents inherit beliefs and activities from groups (representing capabilities, roles, affiliations, work areas, etc.). A subsumption architecture provides flexible perceptual scoping, redirection of attention, and resumption of interrupted activities (Clancey 2002, compares activity and task modeling).

Each Brahms system in the MAA includes three kinds of agents:

1. **Personal Agent:** Represents the interests and activities of corresponding people and systems at some location. For example, the Brahms system running on the ERA includes an agent representing the ERA's beliefs (world view) and activities.
2. **Communication Agent:** A Java-based agent that interfaces between a Brahms system and other hardware or software components. For example, the Dialog Agent interfaces with the speech commanding system provided by the RIALIST group.
3. **Proxy Agent:** Represents agents and objects residing in other Brahms systems (e.g., ATV Brahms includes an ERA proxy agent). Usually redirects communications, but may stand in (e.g., when a mobile agent is out of communication range or unable to respond immediately because of the time delay).

The Brahms models and physical devices are integrated through OAA messages (Martin et al. 1999), KAoS (Bradshaw et al. 1997), and CORBA: An agent in one Brahms model communicates with its proxy agent representation in another model through the KAoS middleware layer. Communication agents serve as interfaces between Brahms and external machines (e.g., the ERA) using CORBA objects.

The total hardware-software system shown in Figure 1 is first *simulated in a single Brahms model*, with the different agents and devices modeled in different locations and carrying out certain (perhaps scheduled) activities, according to the scenario(s) being tested. In addition to the components shown in the ellipses, the people and external systems (e.g., ERA) are also simulated. This simulation, which serves a specification for the structure of the final runtime system, therefore includes:

- Simulated People and Systems (e.g., crew, ATV, Dialog system)

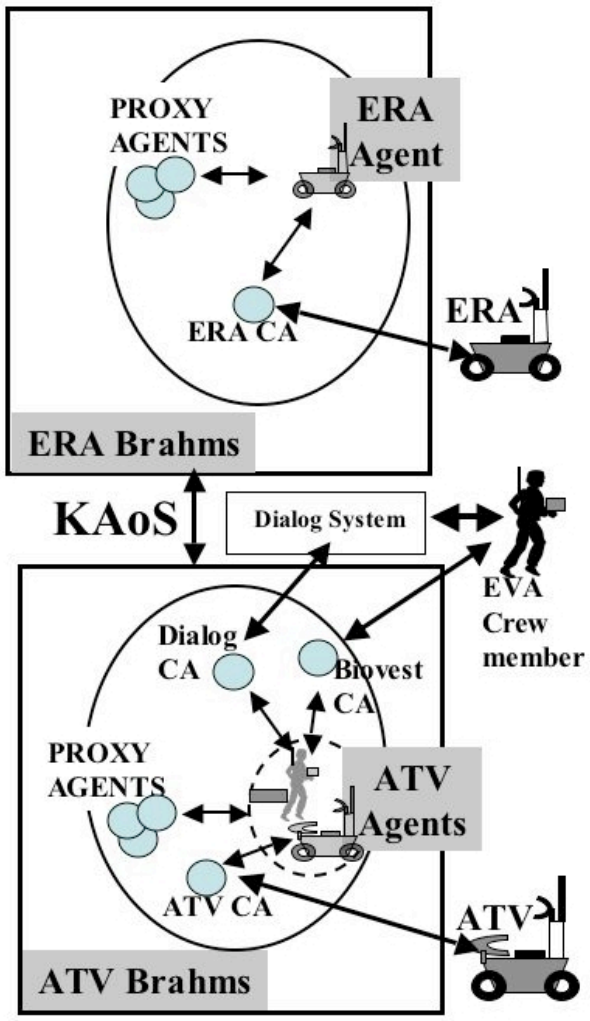

Figure 1: Brahms MAA 2002

- Software Agents (which will become runtime agents)
  - Personal Agents of people (e.g., crew agent) and systems (e.g., ERA Agent in ERA Brahms)
  - Communication agents (e.g., Dialog CA)
  - Proxy agents (e.g., ERA agent in ATV Brahms)

For example, the simulated protocol includes voice commands by a simulated crew member, processed by a personal agent, passed to the simulated Dialog CA, which passes the utterance to the simulated Dialog system. In the implemented runtime system, these components are replaced by actual people and systems (the crew member, the Dialog CA written in Java, and the Dialog system). The personal agents in the simulation remain unchanged in the runtime system.

First Integration Field Test: April-May 2002

The first systems integration field test was completed at NASA/Ames in April 2002. A "level 0" scenario integrated

the Dialog system and a distributed Brahms model running on the MEX ATV and a simulated ERA, using the MEX wireless network. The first field test occurred at JSC at the end of May, including the ERA control software running onboard the actual ERA.

The level 0 scenario demonstrates remote commanding of the ERA by the EVA astronaut, using human speech. The command to the ERA involves taking a still-picture of the astronaut ("ERA take a picture of me"). The Dialog system parses the human voice input and communicates the derived command to the astronaut's personal Brahms agent. The astronaut's personal agent derives the correct robot command (including object references) and communicates it to the ERA's personal agent running onboard the ERA (over the MEX wireless network), which in turn communicates with the ERA using a CORBA interface. The ERA executes the command and stores the image. The ERA's personal agent composes and stores the context of the picture in the form of a picture meta-data object and informs the EVA astronaut that the image has been taken, using the Dialog system.

Besides testing the Dialog/Brahms/MEX interface, another objective of the field test was to apply and test the "simulation to implementation" methodology. This Use-Case method (Jacobson 1994) uses a Brahms simulation model as an agent- and object-based computational functional design of the overall software and hardware system. The *runtime system* replaces the simulated people and systems by their corresponding actual entities. This approach allows us to test a complete MAA system. This is especially valuable for simulating those elements or capabilities that are not still being implemented or only proposed (e.g. new voice commands for controlling the robot).

Second Integration Field Test: Joseph City & Meteor Crater, September 2002

The second field test was held Sept. 3-13, 2002 in Arizona. The goal was to verify the MAA design, implementation, and integration with additional devices in a more authentic field setting. The tests included people wearing an advanced Mark III spacesuit working alongside the ERA (both from JSC), plus a biovest from Stanford University's National Biocomputation Center with physiological sensors, including EKG and respiration, worn inside the spacesuit and transmitted wirelessly to a Personal Data Assistant (iPaq PDA). The speech dialogue system and Brahms were again hosted on MEX, which provided computing and wireless communications on a rugged ATV, proven capable of remote field deployment from previous tests in the High Canadian Arctic.

A variety of functions, available to the astronaut through the integration of physical systems and Brahms agents, were tested using several pre-scripted scenarios involving spoken dialogue:

1. Start and stop gathering and storing biophysical data wirelessly during an EVA. This was tested with and without wearing the Mark III spacesuit.
2. Start and stop tracking the astronaut's and ERA's GPS location data, for recording EVA paths.
3. Providing real-time location information through spoken dialog ("Where am I?").
4. Define places (geographical location objects) in real-time, associating given location names with GPS coordinates for recording visited places during the EVA.
5. Define sample bag objects in real-time, recording when and where samples have been collected.
6. Record voice annotations associated with a time, place, and optionally a sample bag object.
7. Have ERA use its cameras to take photos of the tracked astronaut.

The results of the field test fall into three areas of concern:

1. **Agent architecture:** the system must be better designed to cope with a brittle wireless network, with methods for handling lack of communication, as well as means for people to monitor agent status. The relation between Brahms and the biovest requires separating out low-level sensor processing from interpretation and data archiving (otherwise the Brahms history system can be overloaded with details).
2. **Hardware:** Requires sensors to indicate remaining power and provide warnings; means to handle bandwidth interference and microphone sensitivity; better discipline for configuring connectors, so what is tested is used in the field; augmented ERA capabilities, e.g., to hold a sample bag.
3. **Logistics:** The large number of teams coordinate well, but the next round of tests should eliminate the spacesuit in order to focus on MAA infrastructure; in-situ testing must be better staged (operations were too often driven by including the suited subject, which required special microphones and biovest connections that complicated the simpler connectivity tests that had not yet been accomplished); a more permanent field shelter is required during several weeks of outdoor work.

Somewhat unexpectedly, the most important result was that using a multiagent simulation with scenario-based formal specification greatly facilitated and very likely significantly accelerated cross-institution collaboration. That is, Brahms was shown to be a useful collaborative engineering tool for integrating sensors, automation, procedures, and communication.

Advantages of Brahms Architecture

Developing multiagent systems requires "a systematic means of analyzing the problem, of working how it can be best structured…and then determining how individual agents can be structured" (Jennings et al. 1998, p. 31). The

field tests to date demonstrate that Brahms language and the semantics we attribute to it provide a powerful framework for specifying and implementing a multiagent system. The language formalizes ontological distinctions that lead the system builder to ground agent behaviors in located interactions of explicit *communication* (including voice and written documents), *perception* ("detectables" for forming beliefs conditional on current actions), and *movement* or other physical changes within the environment (e.g., flipping a switch). By enforcing the semantics of language primitives such as "communication," the simulation can be converted more or less directly into a distributed runtime system, in which requests and information are transmitted by actual people and devices, while involving software agents that were previously incorporated in the simulation.

To summarize some of the overall advantages of the architecture that we have drawn upon in creating the runtime systems:

- The language allows seamlessly modeling the interactions of people, robots, devices (e.g., cameras), documents (e.g., a written procedural checklist), communication tools (e.g., a radio walkie-talkie), and any arbitrary physical object (e.g., gloves).
- All people and objects are always located in an explicitly modeled geographic space, which provides important conditions for physical activities (e.g., finding parts, being in close enough proximity to see something, co-location of people leading to informal assistance).
- The ability to convert a simulation into an implementation allows testing a system design in simulated scenarios that model configurations of people, machines, facilities, and geography to be tried in field tests.
- Multiple Brahms systems can communicate and interact via proxy agents, allowing the runtime system to be physically distributed on a wireless network. Thus, existing software components written in other programming languages, such as the Dialog system, can interoperate within the Brahms language of agents, beliefs, activities, and actions.
- The layered communication architecture is robust and flexible, allowing components to be added or removed at runtime: MEX provides a wireless system for transmitting data; KAoS provides a system for agents to register their presence on the network and regulate their interactions through policies. Brahms is used for specifying how agents request or provide data in a proactive, reactive, or responsive way. With appropriate formalization of protocols (e.g., for handling loss of communication), agents can compensate for missing components or leverage the presences of specialized services (e.g., a controllable camera on the ERA).
- Group inheritance of activities and beliefs allows efficiently representing multiple instances of objects and agents (e.g., multiple ATVs). The subsumption architecture implemented by the Brahms engine enables activities to be simultaneously "running" so detectables and conditional actions can handle general goals (survival) and specific tasks (e.g., getting samples) in parallel, while shifting attention by priority-based interruption (i.e., conditional actions called workframes have priorities).
- Brahms' well-developed Integrated Development Environment, compiler, JAVA integration, and Agent Viewer display provides an integrated toolkit for writing, testing, fielding, and reusing a multiagent system. Work in progress involves coupling Brahms to a real-time virtual reality display implemented in Adobe Atmosphere™. For example, this will enable the habitat crew and RST to quickly visualize and hence more easily monitor and control the configuration of the EVA crew, ERA, and other tools.

Related Work

Different research surveys document a wide variety of methods and issues in the development of multiagent systems. Brahms in some respects is atypical, even within this range of research over the past decade. Some of the key differences between Brahms and most of the research reported about multiagent systems are listed here.

- Agents in our current scenarios are not conflicting or competing. Rather the total system is designed for a comprehensive, teamwork interaction that is mission oriented, transcending individual goals. Just as NASA trains flight controllers and astronauts to work harmoniously, a field system incorporating Brahms must have agents that are sensitive to and facilitating the overall activity of the group (of people, robots, and systems). In the Mars exploration environment there will not be "other, un-modeled agents" (Jennings et al. 1998). A "market-based" architecture may be useful for a fleet of robots working together, but the overall system of people and machines must be highly cooperative and coordinated, especially in emergency situations.
- The Brahms tool is designed to produce a runtime system from a simulation. This is both a practical and methodological stance. Developing a simulation is our means of formalizing the requirements of the runtime system. Thus, the description is dynamic, with realizable properties discovered through scenario testing, which drives the entire model-building activity (a methodology we call "empirical requirements analysis", Clancey et al. 2001).
- The architecture shares the decompositional advantages of object-oriented programming, but the semantics of the agent-object distinction in Brahms always leads us to reconsider the differences between people and machines. Specifically, we are building tools to help people deal with complex work situations, many of which are caused by malfunctions or unpredictable behaviors of computerized devices

(e.g., the life support system). Our emphasis is on providing tools for facilitating rather than replacing human judgment ("human-centered computing").

- The explicit modeling of objects, environment, and perception (as detectables) enables including a flexible number of sensors or other instruments in the fielded system.
- The implementation platform enables constructing a multiagent system in which software agents are running on moving, physical systems (e.g., ATVs, astronaut backpacks), hence our interpretation of the phrase "mobile agents."
- We view *autonomy* as a means by which a remote group of people can delegate their monitoring and actions, required because of the communication time-delay and small size of the crew on Mars. For example, a science team on Earth could upload agents to run on the Mars surface, and these agents would make suggestions to the crew or gather data unobtrusively (taking photographs), just as the science team would if they could be on Mars (this is speculative, but surely a highly suitable application for the MAA).
- Our conception of multiagent interactions is that requests are not denied (hence they are more like object-oriented methods [Alonso 2002]). However, agents in different locations with different data may need to exchange beliefs to negotiate a course of action. Or for example, the return of a subsystem to the network (e.g., the ERA) may require other agents to shift from a compensation protocol, to allow more capable agents to complete or redo a task (e.g., making a measurement). The scenarios we have attempted have not required us to confront these issues yet. As we continue our empirical requirements analysis approach of building incrementally and learning about desirable functions in authentic work settings, we expect that some of these complex, theoretical aspects of multiagent interactions may become relevant in the Mars exploration domain.

Acknowledgments

Funding for this work is provided in part by the NASA/Ames Intelligent Systems Program, Human-Centered Computing area, managed by Mike Shafto. As listed in the background section, NASA's Mobile Agents project involves a large team of researchers in several institutions. The work reported here would not have been possible without their ideas and enthusiastic hard work in building the devices and Brahms models we have described. For publications and related information, see http://bill.clancey.name & http://www.agentisolutions.com.

References

Alonso, E. 2002. AI and agents: state of the art. *AI Magazine*, 23(3) 25-29.

Bradshaw, J. M., Dutfield, S., Benoit, P., and Woolley, J. D. 1997. "KAoS: Toward an industrial-strength generic agent architecture." *Software Agents*, J. M. Bradshaw, ed., AAAI Press/The MIT Press, Cambridge, MA, 375-418.

Burridge, R. R., and Graham, J. 2001. Providing robotic assistance during extra-vehicular activity. *Proceedings of the SPIE - The International Society for Optical Engineering*, 4573: 22-33.

Clancey, W. J., Sachs, P., Sierhuis, M., and van Hoof, R. 1998. Brahms: Simulating practice for work systems design. *International Journal of Human-Computer Studies*, **49**: 831-865.

Clancey, W. J., Lee, P., and Sierhuis, M. 2001. Empirical requirements analysis for Mars surface operations using the Flashline Mars Arctic Research Station. *FLAIRS Conference Proceedings*, pp. 24-26.

Clancey, W. J. 2002. Simulating activities: Relating motives, deliberation, and attentive coordination. *Cognitive Systems Research*, **3**(3) 471-499. Special issue on situated and embodied cognition.

Clancey, W. J. in press. Agent Interaction with Human Systems in Complex Environments: Requirements for Automating the Function of CapCom in Apollo 17. *AAAI Spring Symposium on Human Interaction with Agent Systems in Complex Environments*, March 2003.

EVA Robotic Assistant. URL http://vesuvius.jsc.nasa.gov/er_er/html/era/era.html

Jacobson, I. 1994. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Publishing Company, Reading, MA.

Jennings, N. R., Sycara, K., and Wooldridge, M. 1998. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1, 7-38.

Martin D., Cheyer A. and Moran D. 1999. *The Open Agent Architecture: A framework for building distributed software systems*. *Applied Artificial Intelligence*, **13**(1-2), January-March.

Shillcutt, K., Burridge, R., Graham, J. 2002. Boudreaux the Robot (a.k.a. EVA Robotic Assistant). *Papers from the AAAI Fall Symposium on Human-Robot Interaction*, Tech Rpt FS-02-03. Falmouth, MA. pp. 92-96.

Sierhuis, M. 2001. *Modeling and simulating work practice*. Ph.D. thesis, Social Science and Informatics (SWI), University of Amsterdam, SIKS Dissertation Series No. 2001-10, Amsterdam, The Netherlands, ISBN 90-6464-849-2.

Vulkan, N. 2002. Strategic design of mobile agents. *AI Magazine*, 23(3) 101-106.