An Agent Architecture for Comprehensive Mission Robustness

John R. Rose and Michael N. Huhns
Center for Information Technology
University of South Carolina
Columbia, SC 29208
{rose,huhns}@engr.sc.edu

April 25, 2001

1 Introduction

As future NASA missions become more complicated and of longer duration and distance, the software systems controlling them will of necessity have to be large, complex, and increasingly autonomous. We will basically have to trust the systems, so there must be a principled basis for our trust. Unfortunately, constructing large error-free software systems appears not to be achievable by current means. Additionally, the large size of the systems and the unknowns to which they will be subjected cause them to be untestable to even find out if, when, or where they might fail. A new paradigm and architecture for software development are thus needed, and we are investigating ones based on the premise that errors will always be present in software systems, and we should try to not only compensate for them, but also take advantage of them. Furthermore, an agent-based architecture, with the agents having *explicit philosophies*, is a promising foundation for engendering trust. We can trust the agents to act autonomously if they embrace ethical standards that we understand and with which we agree. We expect that this will lead to robustness, fault tolerance, recovery, graceful degradation, and, ultimately, trust in our systems.

2 Software Agent Architecture

A key feature of the architecture is that robustness is achieved through massive redundancy and decentralization in planning and execution. This revolutionary approach to achieving mission goals relies on interaction as a fundamental construct. Competing proposals for satisfying tasks are developed by fluid collaborating groups of software agents vying for resources, so that the failure of one approach does not jeopardize overall mission goals. Competing proposals can be acted on in parallel when they do not interfere significantly with each other, but are pruned as necessary to conserve resources.

This approach to agent behavior will be constrained by a hierarchy of fundamental philosophical principles or societal laws, similar to Asimov's laws of robotics:

- **Principle 1.** An agent shall not harm the mission through its actions or inactions.
- **Principle 2.** Except where it conflicts with principle 1, an agent shall not harm the participants in the mission.
- **Principle 3.** Except where it conflicts with the above principles, an agent shall not harm itself.
- **Principle 4.** Except where it conflicts with the above principles, an agent shall make rational progress toward mission goals.
- **Principle 5.** Except where it conflicts with the above principles, an agent shall follow established conventions.

- **Principle 6.** Except where it conflicts with the above principles, an agent shall make rational progress toward its own goals.
- **Principle 7.** Except where it conflicts with the above principles, an agent shall operate efficiently.

In accordance with these embedded philosophical principles, the system architecture must support common goals, societal laws, common principles, common abilities (such as how to negotiate), and ways to sanction or punish miscreants. The agents will use decision theory in their negotiations to evaluate the expected utility of proposed actions and use of resources. This will result in planning and task execution that is dynamic, rational, massively distributed, occurs at multiple levels of granularity, and can be *trusted*. The architecture is a revolutionary departure, both from current monolithic approaches to planning and from distributed approaches to task execution. It takes advantage of the differences in local perspective that exist in a massively distributed environment to formulate and execute plans that, while not perfect, will not have identical flaws, producing robustness through redundancy and diversity of approach.

3 Programming a Philosophical Agent Architecture

Autonomy and distribution will occur at many levels within the systems. The result we envision will be a programming paradigm where the effort is on assembling and coaching a team to achieve desired functionality. Programmers will specify requirements about what is needed, possibly in terms of social commitments and intentions, rather than specifying how it should be done. The components might negotiate with each other, make social commitments to collaborate, and change their minds about their activities and results.

This vision is based on the premises that (1) *coherence* is more important than consistency, (2) *errors* will always be part of complex systems, (3) error-free code can at times be a *disadvantage,* and (4) where systems interact with the complexities of the physical world, there is a *power* that can be exploited.

4 Conclusion

The agents we construct—and the NASA systems they implement, manage, and enact—must be trustworthy, ethical, parsimonious of resources, efficient, and—failing all else—rational. What we are investigating differs from current work in software agents in that:

- We are not researching new agent capabilities per se
- We are not developing an agent-based system for some new application domain
- We are investigating how agents can be the fundamental building blocks for the construction of general-purpose software systems, with the expected benefits of robustness and autonomy
- We are characterizing agents in terms of *mental abstractions*, and multiple agents in terms of their *interactions*. These abstractions matter because anticipated missions go beyond traditional metaphors and models in terms of their dynamism, openness, and trustworthiness.

The benefit of this architecture to complex missions such as future NASA planetary and deep space missions is fourfold: (1) it will support missions of much greater complexity than are possible under the current model of earth-based control, (2) it will reduce costs by minimizing the amount of earth-based support required for missions, (3) it will essentially eliminate communication time lag as a significant factor in local task execution, providing the ability to react to and take advantage of serendipitous events, and (4) it will significantly enhance mission robustness. The development of the proposed architecture builds on developments in decision theory, agent societies, trusted systems, and ubiquitous computing.