Designing Agents for Systems with Adjustable Autonomy

Paul Scerri and Nancy Reed
Real-time Systems Laboratory
Department of Computer and Information Science
Linköpings Universitet, S-581 83 Linköping, Sweden
pausc,nanre@ida.liu.se

Intelligent agents are starting to assume more responsibility in more critical tasks. Agents do not always work as required, however in some systems it is infeasible to simply "stop" an agent when its behavior is unsatisfactory. Instead it is desirable to reduce the agent's autonomy and give control of the unsatisfactory aspects of the agent's task to a more competent entity. Adjustable Autonomy (AA) is a recent idea that means that the autonomy of agents and humans in a system can vary dynamically. In this paper we look at the relationship between the design of agents and the design of AA for an intelligent system. We present guidelines to help agent designers build agents that are easier to use in systems with AA.

1 Intelligent Agents

Many of the recent, exciting developments in artificial intelligence (AI) have centered around the concept of an *agent*. An agent is an autonomous entity that senses its environment and acts intelligently and pro-actively towards its goals[17; 2]. An important characteristic of an agent is that it has the ability to take actions that affect its environment[7]. Some agents sense and act in purely software environments (e.g. an operating system monitoring agent[16]), while others have a physical embodiment and inhabit a physical environment (e.g. a museum tour guide robot[3]). Because an agent can act, it can be assigned tasks that can potentially be done more quickly, efficiently, cheaply or safely than a human can do them. Thus humans are freed from menial, dangerous and/or boring tasks. Despite the long list of successful agent applications, it is likely we have only scratched the surface of the possibilities intelligent agents have to change the way we live our lives[6; 8]. In the future, autonomous agents will take on more complex tasks and act more intelligently and more decisively.

An *intelligent system* is one consisting of intelligent agents and, possibly, humans and/or other conventional software. Generally in an intelligent system the assignment of responsibility and authority, i.e. autonomy, is either fixed or switches between a small number of fixed configurations. In an intelligent system with AA the system can flexibly configure the assignment of autonomy between the people and agents to best fit the situation.

As AI technology develops, the autonomy agents have increases proportionally, i.e. more capable agents are given more authority and more responsibility. In most cases, once an agent's autonomy is determined by a system designer the agent is left to fulfill its responsibilities according to its specification, within the bounds imposed by its authority and capabilities. In complex environments, an agent will be faced with a vast range of situations and must act satisfactorily in each. However, it is unlikely that any agent has appropriate reasoning mechanisms, sensors and actuators to act appropriately in *all* situations it could potentially face[15]. There will be some situations in which the agent will make unacceptable decisions and, hence, take unacceptable actions – with potentially serious consequences. Some situations are so unlikely that agent designers reasonably ignore them, hence the agent is not designed to handle them properly. Other situations might commonly occur but to build software or robotic hardware to properly handle the situation may be considered too expensive or time-consuming. Yet other situations will be unacceptably handled by an agent due to "bugs" in its software. Importantly, the same reasons that cause an agent to fail to handle a situation satisfactorily, may cause it to not even detect that it has encountered a situation it is not capable of handling satisfactorily. For example, an autonomous robot that cannot detect a wall it should go around, may not detect that it has bumped into the wall either.

The more autonomy an agent has, over more complex and important tasks, the more potentially serious the consequences are when its behavior is unacceptable. However, because the agent is autonomous and because the agent may not detect its own inadequacy, "killing" the agent may be the only way of preventing the incorrect actions. Completely stopping an agent means another entity needs to take over the agent's responsibilities, something that may be unreasonable in a complex physical system (e.g. a spacecraft). A more desirable scenario is if only incorrect *parts* of the agent's activity are taken over while other parts continue to function normally.

Thus, agent developers are faced with a challenging dilemma. For some applications, autonomous agents can be very useful in very many situations, most of the time. But in a small number of situations, a small percentage of the time, agent behavior will be unacceptable and potentially have serious consequences.

2 Adjustable Autonomy

Adjustable Autonomy (AA) is a recent idea meaning to *dynamically change* the autonomy of the intelligent entities, both agents and humans, in a system. Instead of the responsibility and authority of entities being fixed at design time, they can be changed to best configure the system's autonomy to the current situation. The idea is to dynamically assign autonomy to best leverage the constituent entities' strengths and avoid their weaknesses. Thus, *an AA system is an intelligent system where the distribution of autonomy is changed dynamically to optimize overall system performance*. For example, if a human pilot no-

the agent cannot detect, the pilot might like to slightly alter the aircraft's course without taking over all the details of its functioning. Flexible assignment of autonomy means a system can deal with a wider range of situations more effectively. Thus, AA allows the intelligence and autonomy of agents to be fully exploited without being stuck with their inadequate decision making when situations occur the agents cannot handle (or humans could handle better).

3 Conceptual Model of AA

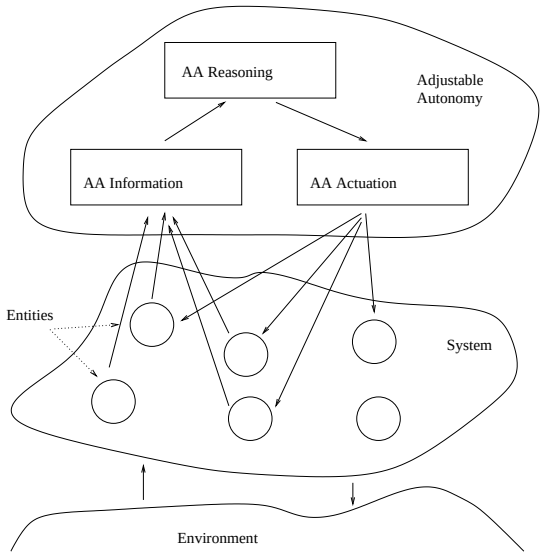

Figure 1: The conceptual relationship between AA and an intelligent system.

A system with AA can dynamically change which entities are responsible for the achievement of which goals by changing decision making responsibility over time. Further, a system with AA can dynamically change the authority constituent entities have to take on particular goals. AA mechanisms manage the changing autonomy by determining appropriate changes in autonomy and implementing those changes.

A key problem to be addressed when building AA is to determine an appropriate distribution of autonomy and provide mechanisms to realize the autonomy changes.

The distribution of autonomy should change according to the current situation and sub-goals, reconfiguring so as to best organize the system resources to achieve the system's goals. Conceptually the task of changing autonomy can be broken into three parts:

- AA Information (AAI) : Collection of the information relevant to the AA decision making.
- AA Reasoning (AAR) : Reasoning about what autonomy changes could or should be made.
- AA Actuation (AAA) : Realization of the decisions made by the AAR.

are shown in Figure 1. AAI provides information on prevailing environmental conditions and the current system state and potential (referred to as *context* in [4]) as are relevant to AAR. The AA reasoner determines what changes in the autonomy distribution will lead to better system performance with respect to its goals. AAR might be done either by a human or in software. Finally, the AAA provides the mechanisms for implementing the decisions of the AA reasoning, i.e. it provides the mechanisms for realizing changes in authority or transfer of responsibility.

The AAI and AAA tightly constrain the design and potential of AAR. Any information not supplied by AAI cannot be used in determination of appropriate autonomy configurations. Likewise, any change that cannot be realized by AAA should not be decided on by AAR. In turn AAI and AAA are both tightly constrained by the services provided to them by the entities in the system. Any information the entities cannot provide cannot be supplied by AAI to AAR. Similarly, any autonomy change the entities cannot accept could not be implemented by AAA. Hence *the services provided by the entities are critically important to the building of an AA system.* Obviously, we are limited to designing the services that software entities provide (as the services of the human entities are fixed).

Consider an analogy to a management consultant at an organization. The consultant's job consists of collecting information, making decisions about organizational changes and implementing those changes. No matter how good the consultant is at their job they rely on employees in the organization to inform them (either implicitly or explicitly) of the current running, goals, etc. of the organization in order to make decisions. If the employees supply limited, insufficient, incorrect or misleading information the consultant's job is significantly harder and their results are likely to be disappointing. Once the consultant makes a decision it needs to be implemented. No matter how good the decision, if it is not accepted and appropriately implemented, the decision is worthless. Implementation of AA works in the same way – if the entities do not supply appropriate information and properly implement decisions, the best AAR is useless.

4 Agent Design and AA

Agent designs differ in the way that information is represented in the agent and how easily it can be extracted in an understandable (to the AAR) manner. Whether the AAI services are simple, just providing access to particular parts of the agent's reasoning, or very complex, needing complex algorithms to extract information from the agent, depends on the design of the agent. The AAI guidelines presented below aim to guide designers to create agents where as much useful, understandable information as possible can be gathered by inspecting the data structures of a running agent. The guidelines try to avoid the need for complex algorithms to extract information from a running agent.

AA Actuation services implement autonomy changes decided on by the AAR. Only those autonomy changes that can be realized by AAA services can be decided on by AAR. Hence, the limitations of the AAA services strictly limit the useful con-

Further, the more elegantly and smoothly the agent incorporates any autonomy changes decided on by AAR into its ongoing behavior, the better the behavior of the overall system. For example, imagine a team of (human) furniture removalists carrying a piano up a staircase. If their foreman yells out from another room that one removalist is being re-assigned to another job he will not (hopefully) just let go of the piano and walk off (leaving his colleagues squashed under a piano at the bottom of the stairs!) but will make the piano safe before moving on. Thus, the behavior of the overall "furniture removal" system is successful because the worker changing tasks switches between tasks in a reasonable manner. The same idea applies to AA where smooth autonomy changes mean better system behavior. If the agent could exhibit similar "common sense" behavior to the furniture removalist the AAR's task is made simpler because changes can be made without (unnecessary) consideration given to transitions of the agent's behavior.

5 Guidelines

We capture our design knowledge of agents for AA systems gained via the implementation of two complete AA systems[13; 10] in a set of guidelines. If followed when designing agents for AA systems, the guidelines should lead to agents with features that makes AA easy to implement.

When designing intelligent agents many competing requirements need to be reconciled. Not all the requirements are related solely to the observable behavior of the agent. For example, there may be particular verifiability, simplicity or computational requirements on an agent[11]. Designers, implicitly or explicitly, follow guidelines when attempting to meet some requirement with a design. A guideline is "a statement or other indication of policy or procedure by which to determine a course of action"[1]. For example, a (simple) guideline to minimize computational requirements for an agent might suggest avoiding algorithms involving significant amounts of search. By following appropriate guidelines designers have a principled, justifiable reason for believing that their design will meet it's requirements. For example, if a design avoids extensive use of search algorithms a designer can argue, with justification, that once implemented the design will be computationally efficient (according to the above guideline).

The guidelines capture our experiences designing agents for AA systems so that other designers can leverage that knowledge. The guidelines given here should be used during the design process when one of the requirements on the development is that the agents will be used in AA systems.

5.1 Designing Agents for AA Information

AA Information services provide the raw information which AAI will present to the AAR. Whatever information AAI services do not provide about the state and intentions of the agent cannot be used in the reasoning. Since the AAR is obviously limited to reasoning about things it knows about the AAI services limit the AAR and hence the AA in general. The details of the precise information needed for AAR (and hence the precise Further, the form that the information needs to be in depend on whether a human or software is doing the AAR.

Agent designs differ in the way that information is represented in the agent and how easily it can be extracted in an understandable (to the AAR) manner. Whether the AAI services are simple, just providing access to particular parts of the agent's reasoning, or very complex, needing complex algorithms to extract information from the agent, depends on the design of the agent. The AAI guidelines presented below aim to guide designers to create agents where as much useful, understandable information as possible can be gathered by merely inspecting the data structures of a running agent. The guidelines try to avoid the need for complex algorithms to extract information from a running agent.

Three guidelines provide advice for designing agents which will lead to easy to build, high quality AAI:

- *Explicit Information Guideline* – Represent the agent's reasoning process and reasoning state explicitly and in a format close to the format that will be used by AAR.
- *Software Engineering Guideline* – Following good software engineering practices in agent design makes it easier to build AA.
- *Design Information Guideline* – Represent information above and beyond the information needed by the reasoning process such that it explains design decisions implicit in the reasoning process.

5.2 Designing Agents for AA Actuation

Being able to change the agent's internal structure, whether indirectly or directly, is the basic goal of AAA as this results in changes in behavior. In theory, if an agent allows sufficient access to it's internal structure an outside entity could force it to do anything that the agent is capable of. I.e., if any part of the agent's internal structure can be changed online then the behavior of the agent could be changed arbitrarily online. However, the nature of the agent's representation of it's behavior affects the *ease* with which an outside entity can dynamically alter the agent's behavior. For example, imagine the difficulty of "reprogramming" the ants building an ant hill to build wide flat hills instead of tall and skinny ones. The new plan would need to be mapped to individual ants and simple pheromone based rules designed – possible but far from trivial. Note that distribution of control is only one element that affects whether the agent's behavior is easy to change. The desire when building AAA is to make the changes required to the internal structures of the agent to effect a particular autonomy change as simple as possible, therefore making the AA as easy to implement as possible.

These guidelines summarize design strategies for building agents whose behavior can be most flexibly and easily changed online:

- *Deterministic Execution Guideline* – Make the reasoning process of the agent as deterministic (and hence predictable) as possible.

plicitly as possible and in a format that requires the least translation.

- *Building Blocks Guideline* – Divide overall behavior into small pieces that are related to each other in a very semantically clear and simple way.

- *No Extra Mechanisms Guideline* – The AAA should not use mechanisms other than the normal reasoning mechanisms used by the agents.

- *Design for Failure Guideline* – The agent should be designed so that if any part of it fails at any time, its behavior will degrade gracefully.

6 Discussion

These guidelines have been followed in the implementation of two AA systems. The first is a system for creating agents for interactive simulations called EASE[13; 14]. In EASE, the AA is used to give a user runtime control over the agents in a simulation[12]. Two domains are being investigated: aircraft control using Saab's air-combat simulator, TACSI[9], and football using the RoboCup simulator[5].

The second implemented system is the E-Elves where intelligent agents are used to help human users carry out every day tasks in a human organization[10]. The agents can look after tasks like rescheduling meetings when a user is delayed, ordering lunch and finding presenters for meetings.

We aim to evaluate the guidelines in the following way. First, we map each of the guidelines to *agent features* in the two systems that come about from following the guidelines. Then we describe the impact of each of the features on how easy it was to implement AA. By showing that agent features encouraged by the guidelines made the AA easy to implement we show the utility of those guidelines. Preliminary results show that following the guidelines when designing agents does make AA easy to implement. Ongoing work is looking in more detail at the relationships between the agent features and AA facilities.

One weakness in this evaluation is that we ask that the reader take on faith that the AA facilities that we show are easy to build are genuinely useful. Proving that the facilities the systems provide are actually useful would require complete implementations of AA (including complete Human-Computer interfaces and AAR) and extensive testing in unstructured environments. Without use in unstructured, real-world environments we cannot say the facilities are genuinely useful – any experiment we create will be "rigged" to the facilities we provide. However, such testing is beyond the scope of this work. In fact, showing the real world usefulness of AA presents an interesting problem in general. AA is fundamentally used for doing things not anticipated by the software (i.e., the software designer). How can we (as designers) do useful, repeatable, controlled experiments on the ability of software to do things we (as designers) did not expect!? The only really effective tests of AA will come through extensive real-world use of the system. The best we can do is point out that the literature offers much motivation for AA with AA would be genuinely useful in the real world.

In some cases, due to other design constraints, we have violated the AA guidelines when designing the agents. In such cases, we can look directly at the implications of violating the design guideline on the implementation of AA. Preliminary evaluation shows that in general, when we violated the guidelines from Section 5, AA facilities were more difficult to build on the resulting features. By showing that violating the guidelines made AA more difficult to implement we can further strengthen the case for the utility of the guidelines.

Clearly, when designing a particular agent there are many things to take into account other than making AA easy to implement. Most obviously, the agent must be able to achieve the behaviour required of it. Other design requirements like the ability to formally verify the agent's behaviour will also be made simpler or more difficult by the design of the agent. Guidelines, either explicit or implicit, will advise ways to meet other requirements in a straightforward manner. In the usual case it will not be possible to design an agent which makes everything straightforward, i.e. it will not be possible to follow the guidelines for all the required agent features. Whenever the AA guidelines are violated, due to conflicts with guidelines for other requirements, AA will be more difficult to implement. Such "tradeoffs" are no different than the tradeoffs between, say, efficiency and flexibility, where improving one will often be to the detriment of the other. Even in our implemented systems (where AA was a very high priority) AA guidelines were sometimes violated so other design requirements could be met. However, despite the need to violate them occasionally, guidelines are still useful because they allow the designers to make *informed* tradeoffs about competing requirements. Hence, the guidelines allow the designer to understand early in the design process that a certain design decision is making AA more difficult to implement which in turn might lead them to explore other options.

It is important to note that the agent designs for the two implemented AA systems are distinctly different. EASE agents have a reactive, behavior based style architecture while E-Elves agents use a model based, decision theoretic approach. The fact that there is a distinct difference is important because it shows that the guidelines are not too restrictive, in that they permit a variety of different designs provided certain characteristics are present. The guidelines require certain *features* of an agent design but do not overly constrain *how* those features are achieved. Because the applications of AA systems are going to be very diverse it is critically important that the guidelines do not overly restrict the designer's options. Consider an analogy with a guideline for making comfortable shoes which advises using a hard material on the bottom (to protect the foot) and a soft one on the top (to give flexibility). This guideline is met in a huge variety of different shoe designs with other requirements, e.g., football boots, dress shoes, running shoes, hiking boots, etc. However, in some cases other requirements on the shoe's "performance" may mean the guideline is violated, e.g., for astronaut space walking boots. When the guideline is followed comfortable shoes result and when violated uncomfortable shoes result. We believe that the AA guidelines we

designs all with the property that AA is subsequently straightforward to implement.

Acknowledgments

This work is supported by The Network for Real-Time Research and Education in Sweden (ARTES), Project no. 0055-22, Saab Corporation, Operational Analysis Division, the Swedish National Board for Industrial and Technical Development (NUTEK) under grants IK1P-97-09677, IK1P-98-06280 and IK1P-99-6166, and the Center for Industrial Information Technology (CENIIT) under grant 99.7.

References

[1] *The American Heritage Dictionary of the English Language*. Houghton Mifflin Company, 1996.

[2] Jeffery Bradshaw. An introduction to software agents. In *Software Agents*, pages 3–49. MIT Press, 1997.

[3] Wolfram Burgard, Armin Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. The interactive museum tour-guide robot. In *Proceedings of AAAI'98*, pages 11–18, 1998.

[4] H. Hexmoor. Case studies of autonomy. In *Proceedings of FLAIRS 2000*, pages 246–249, 2000.

[5] Itsuki Noda. Soccer server: A simulator of RoboCup. In *Proceedings of AI Symposium'95*, Japanese Society for Artificial Intelligence, December 1995.

[6] B. Pell, E. Gamble, E. Gat, R. Keesing, J. Kurien, W. Millar, P. Nayak, C. Plaunt, and B. Williams. A hybird procedural/deductive executive for autonomous spacecraft. In *Proceedings of the second international conference on autonomous agents*, pages 369–376, 1998.

[7] S. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.

[8] P. Rybski, S. Stoeter, M. Erickson, M. Gini, D. Hougen, and N. Papanikolopoulos. A team of robotic agents for surveillance. In *Proceedings of the fourth international conference on autonomous agents*, pages 9–16, 2000.

[9] Saab AB, Gripen, Operational Analysis, Modeling and Simulation. *TACSI - User Guide*, 5.2 edition, September 1998. in Swedish.

[10] P. Scerri, D. Pynadath, and M. Tambe. Adjustable autonomy in real-world multi-agent environments. In *Proceedings of the Fifth international conference on autonomous agents (Agents'01)*, 2001. to appear.

[11] P. Scerri and N. Reed. Engineering characteristics of autonomous agent architectures. *Journal of Experimental and Theoretical artificial intelligence*, 12(2):191–212, 2000.

[12] Paul Scerri and Nancy Reed. Demonstrating AA in real-world multi-agent environments. In *Technical Abstracts of Technical Demonstrations at Agents 2000*, 2000.

[13] Paul Scerri and Nancy Reed. Creating complex actors with EASE. In *Proceedings of Autonomous Agents 2000*, pages 142–143, 2000.

[14] Paul Scerri and Nancy Reed. The EASE actor development environment. In *Proceedings of the Workshop of the Swedish AI Society, SAIS'2000*, 2000.

[15] B. Shneiderman. *Designing the User Interface*. Addison Wesley, 1998.

[16] Hongjun Song, Stan Franklin, and Aregahegn Negatu. Sumpy: A fuzzy software agent. In *Proceedings of the ISCA Conference on Intelligent Systems*, pages 124–129, Reno, Nevada, 1996.

[17] Michael Wooldridge and Nicholas Jennings. Agent theories, architectures and languages: A survey. In *Intelligent Agents*, pages 1–32. Springer-Verlag, 1994.