Behaviors of Actors in a Resource-Exchange Model of Geopolitics

Curtis S. Cooper[1], Walter E. Beyeler[1], Jacob A. Hobbs[1],
Michael D. Mitchell[1], Z. Rowan Copley[1,2], and Matthew Antognoli[1]

[1] Sandia National Laboratories,
Albuquerque, NM, 87185-1137 USA
[2] St. Mary's College of Maryland
{cscoope,jahobbs,webeyel,micmitc,mantogn}@sandia.gov,
zrcopley@smcm.edu

Abstract. We present initial findings of an ongoing effort to endow the key players in a nation-state model with intelligent behaviors. The model is based on resource exchange as the fundamental interaction between agents. In initial versions, model agents were severely limited in their ability to respond and adapt to changes in their environment. By modeling agents with a broader range of capabilities, we can potentially evaluate policies more robustly. To this end, we have developed a hierarchical behavioral module, based on an extension of the proven ATLANTIS architecture, in order to provide flexible decision-making algorithms to agents. A Three-Layer Architecture for Navigating Through Intricate Situations (ATLANTIS) was originally conceived for autonomous robot navigation at NASA's JPL. It describes a multi-level approach to artificial intelligence. We demonstrate the suitability of our reification for guiding vastly different types of decisions in our simulations over a broad range of time scales.

Keywords: Complex systems, software architecture, agent-based simulations, artificial intelligence, robot planning, emergent behaviors, policy analysis.

1 Introduction

There is growing interest in the social sciences in numerically modeling the key processes and interactions important for determining change in geopolitical systems. A major long-term goal is to develop models to critically evaluate U.S. foreign policy in a complex arena of multi-national corporations, military and economic rivalries, and long-term changes in the balance of power among nations. Eventually, the most useful models will capture the breadth of values and ideologies that guide nations of various sizes and political configurations.

The resource-exchange model of Beyeler et al. (2011) [1] has been adapted to be capable of representing both historical and contemporary economic interactions among key players in the geopolitical arena. The agents in the model can include national and sub-national entities (such as provinces) as well as corporations, whether

K. Glass et al. (Eds.): COMPLEX 2012, LNICST 126, pp. 70–82, 2013.

sponsored by a single nation to advance its interests (e.g., the East India Company) or a semi-autonomous organization with international affiliations (e.g., Microsoft). We often refer to the model as the "nation-state model" in this configuration, which is well-suited for geopolitical scenarios. Both terms, "resource-exchange model" and "nation-state model", will be used interchangeably throughout the text. Although it does not yet include prediction of military conflicts and their outcomes, the model nevertheless captures essential features of real economic systems. For example, it has been used successfully to represent economic interactions in the current Pax Americana, in which global-scale military conflicts are inhibited.

The nation-state model [1] is a hybrid model combining system dynamics and multi-agent based modeling. Differential equations representing the important rates of change in the system are integrated forward in time. The continuous timeline described by the rate equations is interspersed, however, with discrete resource-exchange events between agents (which the authors call entities). These exchanges occur through markets and constitute the fundamental interaction between entities. Money is represented in this framework as simply a resource which can be exchanged for any other, whereas all other resources are exchanged in specific ratios according to their relative value to different suppliers and consumers. Whether corporate, national, or sub-national, entities in the nation-state model [1] instantiate new processes and change parameters of processes already in operation in order to affect the flow of resources in and out of their boundaries. Processes represent what entities *can* do; what they *will* do is a separate question.

It is noteworthy that entities in the current implementation of the exchange model [1] do not have the ability to make choices or analyze their environments (e.g., to improve their performance). Their ability to adapt is limited to changing their valuations of different resources according to simple dynamical equations. Entity behavior is governed solely by the dynamics, which do not represent goal-seeking, internal world-model building, or any other intentional process. The behaviors exhibited are therefore quite limited. In assessing the strength of foreign-policy decisions, naïve models carry the risk of underestimating the capabilities of adversaries and therefore evaluating U.S. policies over-optimistically. Geopolitical players in credible models will be capable of innovative approaches, such as trade regulations and the initiation of conflicts, to promote their own goals and undermine those of their competitors.

It is the goal of this effort to implement agent behavior (with varying levels of skill) for the resource-exchange model using ideas from the artificial intelligence community. Our approach has been to design and develop reusable Java language [2] modules, which themselves depend minimally on the structure of the resource-exchange model, in order to facilitate future application of the behavioral layer to other agent-based simulations. Architecturally, the software is a reification of the ATLANTIS architecture developed by Erann Gat (see [3], [4]) as a solution to the complex problem of navigating NASA's ground rovers through rocky Martian terrain. We will discuss the advantages of their design for guiding entity behavior in the nation-state model.

2 Intelligent Agents

2.1 Degrees of Sophistication

Defining intelligent decision-making processes by agents in complex systems is critical to understanding the interactions between them, which can be both cooperative and competitive.[1] The algorithms that accomplish this are expected to be broadly applicable. By not tying the behavioral layer to a particular complex-system model, we confer the ability for it to be used to inform decision-making for a wide range of agent-based systems. For example, flocking and steering behaviors can be encapsulated as common algorithms of certain animals. Once a particular behavior for an animal has been implemented, other animals exhibiting similar behaviors can be modeled using existing algorithms as a foundation.

Beyond intelligent agents, it is desirable to be able to provide certain agents with decision-making capabilities that are not cognitive, knowledge-based, or even particularly intelligent. For example, to assess the performance of a particular entity and its decisions, it may make sense to script the specific actions of its competitors. Similarly, in systems such as ant colonies, inter-agent cooperation is possible without cognitive decision-making by any one member. Individual ants behave reflexively to specific chemical stimuli they receive from others in the colony. Such autonomous agents are well-adapted to their problem domain and achieve success without manifesting human-like thought processes.

The first-order, rudimentary approach to modeling behavior, which has been used with some success in video games, is scripting. Scripted actors have a set strategy they follow. For example, a nation-state in our model could be told to always spend 10% of its GDP on military products, unless that action engenders a response from others. This approach, which is straightforward to implement, can certainly instill nations and corporations with behavior. It is important to recognize, however, that scripted, predictable strategies, in which the behaviors of agents cannot adapt in time to new input and new situations, will almost certainly fail in competitive environments against adversaries that learn and adapt.

The next level of sophistication for agent behavior is to instill actors with reflexes. For example, even animals we would not consider to be particularly intelligent have the ability to blink an eye to avoid being hit by a pebble. This rule-based approach to behavior is sufficient for making good decisions in some situations. For example, contemplation is not required (or a good idea) to avoid a car crash while driving. Braking and/or steering actions must be taken in a timely manner to preserve the driver's health.

The most sophisticated artificial intelligence systems, designed to solve challenging problems, are deliberative in nature. That is, the decision-maker must evaluate multiple options and make choices based on imperfect or distorted information about its surroundings, making outcomes of certain choices very difficult to predict. Although challenging to implement (and potentially intractable),

[1] The ideas we present in this section come from internal discussions as well as a review of the artificial intelligence literature, especially the excellent textbooks that have been written on the subject; see [5] and [6].

deliberative decision-making may be needed in order to achieve the ultimate goals of this research: to begin evaluating computationally the performance of diverse policy choices in a multitude of hypothetical scenarios.

2.2 Components of Intelligent Agents

We assume here that the agents we will be dealing with in practice are neither omniscient nor omnipotent. They are limited in their knowledge (and their ability to acquire knowledge), and those agents that can perform actions have a discrete, limited set of capabilities to affect their surroundings. Some but not all agents will also have memories and the ability to learn from past experiences. For agents that learn and adapt to their surroundings, we do not expect repeatable output for a given sequence of inputs. Intelligent agents exhibit dynamic behaviors; they are capable of improving their performance through repetition.

We consider as a reasonable starting point a functional model for decision-making, which has the flexibility to encapsulate at least some common elements of the range of problems that require agents to make decisions and perform actions. Note we conceive of decisions and actions (or plans of action) as separate concepts. The latter are clearly the output of any decision model, but the implementation of a plan is time-dependent and therefore susceptible to dynamical obstacles in the environment or disruption by the actions of other agents. In other words, things don't always go according to plan. Dynamic, intelligent agents should be capable of perceiving obstacles and rethinking plans as unforeseen circumstances arise.

Before discussing our proposed functional decomposition of decision-making problems, other temporal questions should be mentioned. It seems likely that complex agents will perform multiple tasks simultaneously. How often do decisions about these tasks need to be reconsidered? What level of analysis or effort is appropriate for each decision type? How much time is available to make each decision before action is required for the well-being (even survival) of the agent? How is the timing of decisions related to the rate of sensory input and the agent's ongoing accumulation of knowledge about its environment? Answers to these questions are likely to be highly application-dependent. We note here only that how decisions are made is distinct from when decisions are executed. An application attempting to model the behaviors of intelligent agents must consider both problems in turn.

If the output of a decision-making model (or algorithm) is a plan of action, what is the input? To determine this, we must first separate elements of the problem intrinsic to the agent from extrinsic elements. The latter are the agent's environment, from which it receives input through its sensors and receptors in the form of *percepts*. Environmental awareness is limited by the agent's sensors (which poll for new percepts) and receptors (which receive signals from the environment). The time scales for updating the agent about the environment depend on how these sensory mechanisms operate (e.g., ears and eyes, through which animals gather information at different frequencies).

The agent must have a self-model of its sensors, actuators, and world view, which specify the inputs to various decision-making problems. Note also that complex agents may have different approaches to selecting actions in different situations. For example, human eyes blink reflexively to prevent damage to them. Such low-level

tasks can be addressed effectively by quick responses to reflexive impulses. Humans are also capable, however, of solving difficult problems through knowledge-based, cognitive thought processes.

Hence, for agents that acquire and accumulate knowledge, decisions are not always made in reaction to percepts directly. Rather, sensory information is acquired, filtered, processed, and stored in the form of memories. Cognitive decisions are made based on knowledge–i.e., some sort of world model–comprising an agent's fragmented memories about its environment and its experiences with the outcomes of past decisions. Certain percepts to be sure (such as an imminent car crash) will trigger immediate decisions at specific times. Intelligent agents, however, will also make decisions at various (in principle unpredictable) times, depending on the importance of the topic, the agent's self-confidence about its current plans, and the availability of new, relevant information.

2.3 Decision Problems and Thinking Agents

As mentioned in Section 2.2, we hypothesize in this discussion that many important decision-making problems can be embodied in a *decision function*. The decision function can be represented abstractly with a decision model, which evaluates different actions against whatever (presumably domain-specific) criteria are appropriate for the problem at hand. From an object-oriented programming (OOP) standpoint, the decision model is likely to be an aspect of the agent attached using some sort of strategy pattern [7]. This design makes it possible for new ways of making decisions (for different types of problems) to be dynamically plugged into the agent. We elaborate here on the rationale for the simplified form of the decision function that we have adopted in these interfaces.

The decision function clearly must take in the sequence of percepts from the agent's sensors that have been acquired since the last time the agent processed sensory information (e.g., by acting on it or incorporating the new information into its world model). In addition to the environmental information (what the agent knows or can detect), we also will need a description of the agent's actuators; i.e., those things that the agent can do. A plan of action, which is the output of a decision function, will naturally consist of a sequence of actions within the set of actions possible (e.g., according to the agent's physics). The set of allowed actions limits the scope of each decision problem to a finite set of possible choices, although the correct plan for the agent might in some cases consist of an infinitely repeating sequence of actions.

With suitable time-dependent decision functions, engineered agents in models can mimic intelligent behavior (at least in specific problem domains). Agents must combine their knowledge to solve problems in their task environments, with the potential to improve their performance with experience. In practice, decision-making algorithms can consider multiple choices for actions (or plans of actions) against various goodness criteria. The challenge is then optimizing the agent's path through the decision tree, considering tradeoffs and likely events in the environment and responses from allies, neutrals, and adversaries.

Significant progress can be made towards self-aware, thinking agents using a simplistic model of human psychology, the so-called *rational-actor* hypothesis. Rational-actor models assume intelligent agents always pursue their own interests, or

what they believe to be their own interests based on past experiences, limited only by their imperfect ability to predict the future outcomes of their actions. That is, they are goal-oriented and seek effective solutions to the problems they encounter to accomplish their goals. The filtering of information by sensory mechanisms in the agent leads to skewed perceptions or incomplete knowledge about the system.

The rational-actor hypothesis may be flawed (in that a rational actor will not always mimic human behaviors, which are not strictly rational), but it nevertheless affords great breadth of freedom to entities in a framework such as the nation-state model [1]. The notion that judgments are subjective can be captured by differences in values between entities; i.e., those concepts the entities view as priorities. The value sets can differ greatly between entities and are in principle time-dependent, varying with a given entity's perception of its surroundings. In rational decision-making, all options for an entity to pursue its interests are on the table.

3 The ATLANTIS Agent

3.1 Description of the Architecture

Many approaches have been tried since the inception of artificial intelligence to address the complexities involved in programming agents to solve problems (for example, see the discussion in [8]). In perusing various proposed solutions in the literature, we came to the conclusion that a multi-layered approach would provide the flexibility sought for governing the behaviors of entities in the nation-state model, for reasons we discuss in detail in Section 3.2.

We eventually settled on the three-layered architecture described by ATLANTIS and shown in Figure 1. We also show in Figure 2 the key interfaces in our Java reification of ATLANTIS derived from an object-oriented design process, and how the agents interact with the other objects in the system. In particular, state information about the world is encapsulated as “projection” objects. The intent of Figure 2 is not to befuddle the reader with unnecessary implementation details but to show how the ATLANTIS architecture translates in practice to a statically typed language like Java (note Gat’s original implementations of ATLANTIS were written in LISP; see [3], [4]).

The primary attraction of the three-layered approach for guiding agent behavior is its modularity. By dividing the decision-making process of an actor (our term for the subclass of agents that make decisions) into three separate layers—control, sequencing, and deliberative (see Figure 1)—the architecture supports implementations of increasing (and ultimately arbitrary) levels of sophistication. We will first describe the control layer and deliberative layer, which are the most intuitive conceptually.

The control layer defines all interactions with the actor’s task environment. Actions must be performed by specific actuators attached to the agent, and actuators can only perform one task at a time. This is a key constraint that greatly simplifies the system and allows for local processing of information by the actor. As the only tools available to actors to affect their environment in constrained ways, actuators greatly

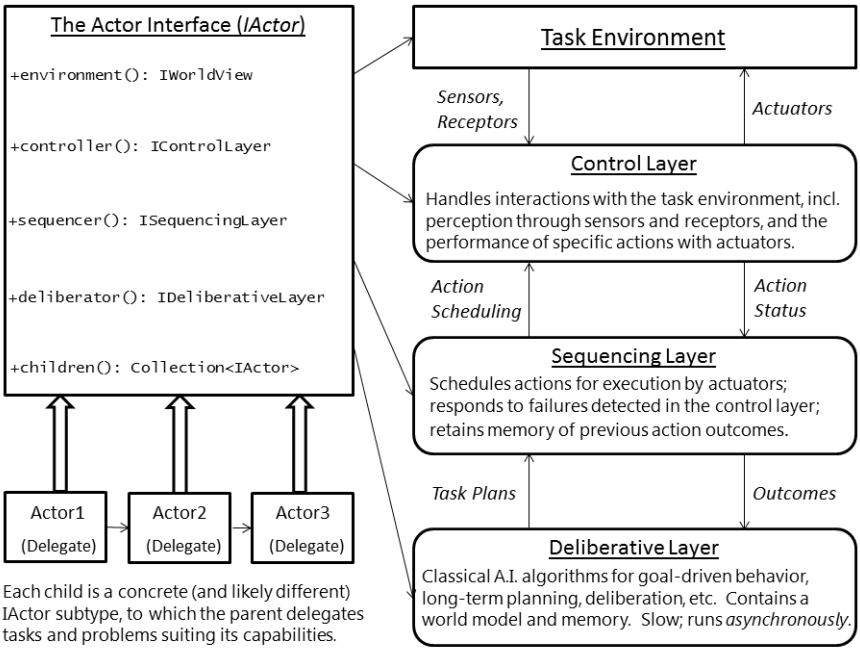


Fig. 1. As shown in this high-level overview of the ATLANTIS architecture [3], actors aggregate three interacting layers to make decisions and accomplish tasks. The control and deliberative layers send messages to and from the mediating sequencing layer to coordinate tasks. The control layer is close to the environment and operates on short time scales, whereas the deliberative layer, which runs asynchronously, solves complex problems and suggests action sequences for queuing by the sequencing layer.

limit the scope of the agent, the decisions it must make, and the information required to make those decisions. Reflexes by actuators on short time scales are possible in the control layer (e.g., blinking eyes), thus permitting basic agent behaviors. At this level of the system, however, the actor does not manifest "intelligence" (or problem-solving). Rather, it performs only the low-level, primitive actions defined by its actuators.

Note in Figure 1 that an actor's 'environment()' method returns a "world view" object, not the world itself. The task environment (for the most general types of acting agents) is itself a function of the agent's perceptions. Actors interact with an imperfect model of their local environment; they do not know the full state of system. Any information actors have about their surroundings must go through their sensors or receptors. They have no access to other information about the world. This "fog" is another key constraint simplifying the construction of a true, problem-solving agent: they are not omniscient and require specific tools to gather and process information.

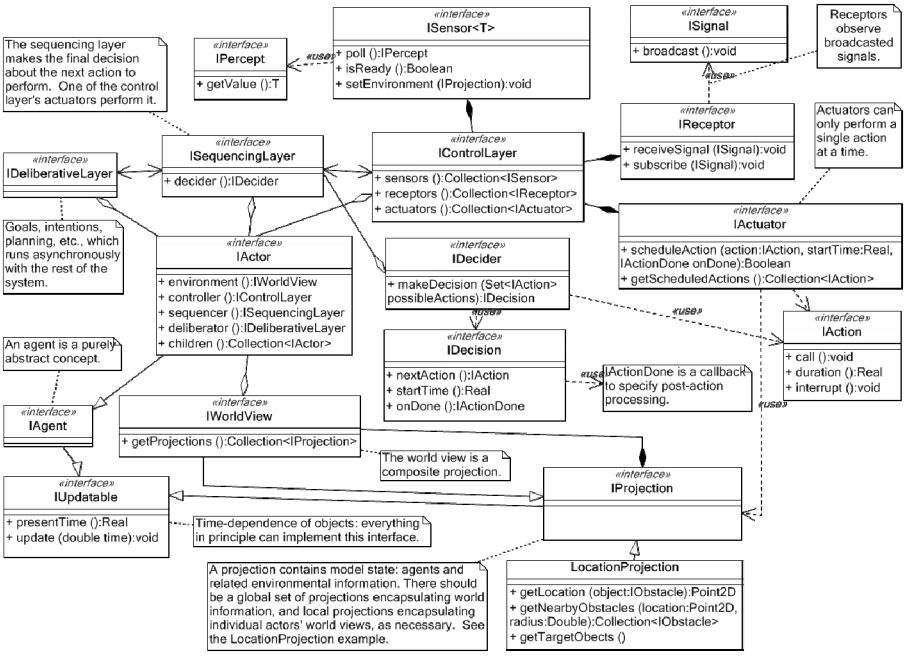


Fig. 2. In this class diagram (see [7] for more information about UML) of the key objects involved in our Java-based reification of the ATLANTIS architecture (for intelligent decision-making), the task environment is represented with composite projections (i.e., IProjection implementations). We represent the notions of perceptions and beliefs with these projection objects. Hence, the world view of an actor can represent a distorted or incomplete description of the state of the system. All direct interaction of the actor with its environment occurs through the control layer.

In ATLANTIS, the most sophisticated problem-solving for the actor occurs in the deliberative layer, which notably runs asynchronously with the rest of the system, so it does not reduce the responsiveness of the control layer. The bulk of the memory and processing power of the agent is expected to be consumed by deliberation, which includes development of a world model, time management, action selection and planning, and achieving goals. The learning also occurs in the deliberative layer by retaining memories of past experiences. In a robotics system, the control layer is a robust, real-time simulator, whereas the deliberative layer is not real-time but rather must partition its computational resources to achieve long-term goals prioritized by their importance to the agent.

The sequencing layer is the core innovation of ATLANTIS and three-layer A.I. architectures in general (see Gat, 1998 [8]). By allowing agent control and deliberation to operate asynchronously, with the controller always having real-time priority, the agent can potentially perform both simple and complicated tasks well. In early intelligent-agent architectures, the deliberator was in complete control. Such an architecture is much more limited because action selection is often only occurring on

long timescales compared to the polling time of a typical agent's sensors (e.g., of order 10 Hz for the human vision system).

The sequencing layer is in principle complicated. It must sort out the priorities of different actions, provide scheduling, interruption, and rescheduling of tasks as necessary, and respond to failure messages sent by the control layer. For simple agents, this is likely to be fairly straightforward. Difficulties emerge for more complex agents, however, in which goals and plans can potentially conflict or compete for use of actuators. Much complexity is hidden in the sequencing layer, which we consider a good design. By delegating final authority for action scheduling to a mediating layer, a good implementation can allocate computational resources appropriately and balance the agent's short-term and long-term needs.

Note another key feature of ATLANTIS shown in Figure 1: the hierarchical structure of the actors themselves. In object-oriented design terms, actors are naturally organized in a composite pattern [7]. Hence, specific tasks either too complicated or too low-level for the agent can be delegated to a set of child actors (the delegates in the diagram) under the supervision of the parent. Very different implementations might be used for decision-making by an actor's children, allowing the intelligent agent to solve multiple specialized problems at once. For instance, a nation-state might delegate discrete information-gathering to an espionage agency. To make life interesting, child actors may not always act in the best interests of the parent (depending on the level of oversight).

Finally, we point out that the control and sequencing layers together, without a deliberative layer, are sufficient to capture certain types of agent behaviors, such as reflexes occurring over short time spans, in which deliberation and planning are not necessary to make reasonable decisions. Initial implementations should therefore focus on constructing a robust control layer supervised by a simple sequencing layer, which does not need to do much work because actions are simple and the time scales for their completion are short. Such a configuration can exhibit behaviors based on simple rules, such as an expert system. Behaviors in such a system can be quite interesting in their own right (e.g., emergent behaviors observed in cellular automata), although complex problem-solving and learning still require the deliberative layer.

3.2 Strengths and Weaknesses

The solution offered by the ATLANTIS architecture offers numerous advantages over many others we investigated (e.g., a blackboard architecture (briefly described in [5], p. 369-70). A full Java implementation of the components (even ignoring domain-specific objects and/or algorithms) is likely to involve a lot of code. But the architecture affords a sufficiently fine-grained division of labor amongst the components (without sacrificing flexibility) to allow the system to be built from the bottom up. Bottom-up construction is highly desirable because the individual pieces of the software are reasonably simple and testable. Furthermore, simple agent behaviors can be examined in detail before attempting to address challenging domain problems.

The ATLANTIS architecture stays close to the objects in the problem domain itself and is therefore more intuitive than many other architectural descriptions for intelligent agents. It lends itself to development of a Java version without having detailed knowledge of the original version (or access to its source code, which was written in LISP by Erann Gat while at Virginia Tech and NASA's JPL; see [3, 4]).

The architecture also strongly decouples the parts of the problem that are challenging (deliberation in order to solve domain-specific problems) from aspects that are more straightforward (agent control through sensors and actuators). In software-engineering terms, ATLANTIS describes objects with weak coupling but strong cohesion, which is a key principle of building maintainable software (as discussed at length in [7]). This decoupling of deliberation and control extends not just to code complexity but computational resources, since the deliberative layer runs asynchronously from the rest of the system and is allowed to consume more memory and processor time than the other layers.

We furthermore consider it a major advantage of ATLANTIS that dealing with error conditions and reporting unknown situations is well-specified at the architectural level. It is the control layer's responsibility to detect the end state of the actions it attempts (through actuators) and report this information to the sequencing layer. Agents in ATLANTIS are *failure cognizant* [4]. This implies that in a fully functional system, there is natural cohesion between the information-gathering objects (sensors and receptors) and the effector objects (the actuators).

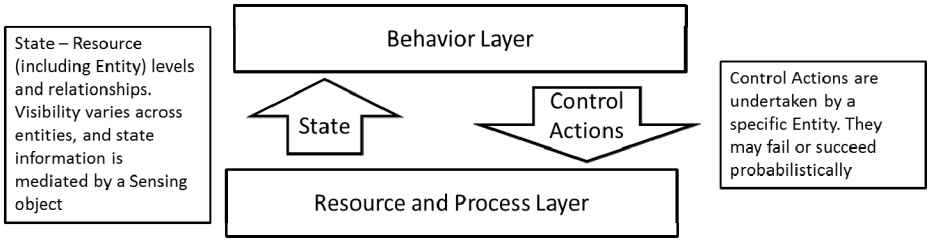
It is worth noting that the intermediate sequencing layer is potentially complicated, depending on the application. Indeed, part of the architecture's design is to hide complexity in the sequencing layer. However, for a first pass-implementation, its operation can be approximated with a thread-safe priority queue (thread-safe because the deliberative layer runs asynchronously), in which the control layer has the option to interrupt in-progress actions at any time. We admit, however, that the sequencing layer is the main potential disadvantage to three-layered architectures. For complicated agents, it is conceivable that a robust implementation of the sequencing layer will prove elusive.

Nevertheless, a solution based on ATLANTIS has been deployed to solve a sophisticated robotics problem: how to autonomously steer a robot on the surface of another planet. Hence, the basic ideas are thought to be sound and compare favorably to alternative approaches to intelligent agent architectures (see [7]). Due to the generality of ATLANTIS and its dissection of the key interacting objects into manageable pieces, we expect the architecture to have broad applicability to complex systems problems, even though many of the anticipated applications are quite different from the robotics problem domain in which it was conceived.

4 Application of ATLANTIS

We have used the ATLANTIS architecture in designing the behavioral layer for entities in the nation-state model. As shown in Figure 3, it is within the behavioral layer that entities process information and perform actions, thereby manifesting complex behaviors that potentially can defy prediction. Without the behavioral layer, behaviors of entities are limited to simple local functions of their internal resource levels [1]. The inclusion of the behavior layer provides entities, such as nations and

corporations, with capabilities to alter model structure in more sophisticated ways in order to promote their policies and weaken competitors. Implementations will enumerate these possible actions (examples of which are shown in Figure 3), from which decisions will be made according to different strategies and algorithms encapsulated in the behavior layer (see Figure 4).



Kinds of Control Actions	
Concrete Description	Abstract Description
Instantiate a new process	
Connect to/Disconnect from a Market	Add/Delete a relationship between an Entity and a Market Entity
Instantiate Entities	Trigger a specific kind of production process
Adjust parameters on processes	
Join/Leave a compound Entity	Add/Delete a relationship
Set defense policy	Change boundary rules
Collect information	Receive messages (of a specific kind)

Fig. 3. The connection between the behavioral layer of a resource-exchange model [1] entity (configured for geopolitical scenarios) and the model structure itself, comprising the external environment with which entities interact. Some examples are given in the table of the types of actions entities must choose from in order to succeed and increase their health.

Sensations and capabilities of entities, shown in Figure 4, as well as discrete action execution by actuators, comprise the control layer of the ATLANTIS architecture. The control layer is the hard boundary between entities and the world. This structural locality is built-in intentionally to confine the scope of entities' knowledge, awareness, and ability to affect their surroundings. Entities are part of the world (and may understand it structure) but only act locally. Internal thought processes of the most intelligent entities (e.g., national leaders) might include a world model with memory, values and objectives, contemplation of future possibilities, and action planning in order to accomplish goals. The black box in Figure 4 captures these ideas of scoring choices, which are made in the deliberative layer. Final arbitration of the actions to be performed by an agent when its goals or needs are conflicting is handled in the sequencing layer.

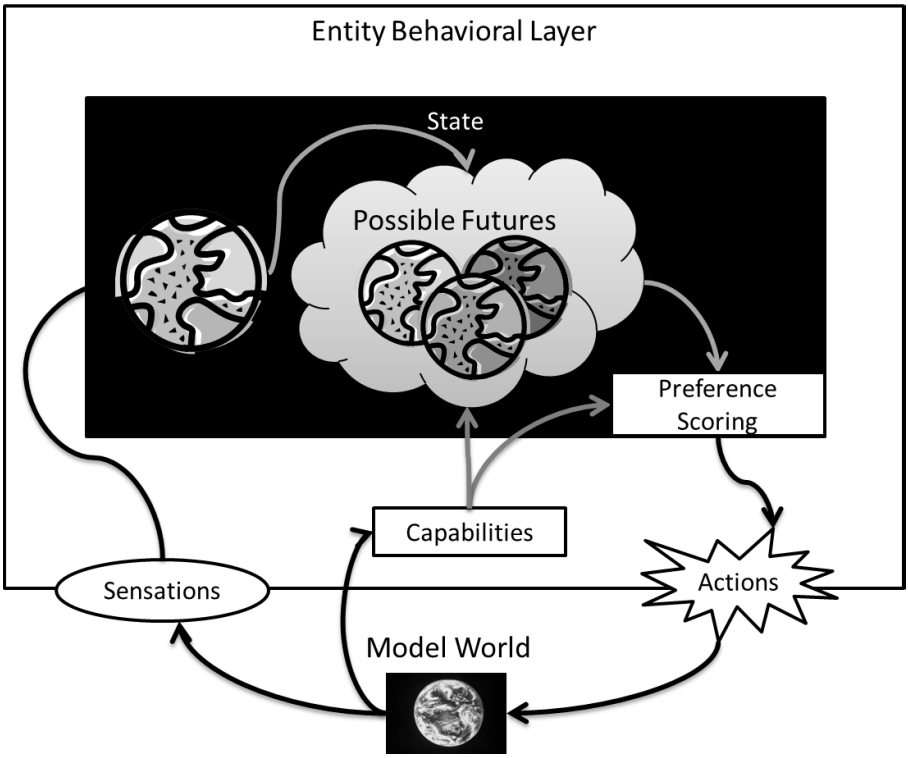


Fig. 4. The behavioral layer will provide intelligent decision-making capabilities to entities in the nation-state model [1]. We discuss in the text how these components fit into our reification of the versatile three-layer architecture (ATLANTIS) in Java, as shown in Figure 2.

The ATLANTIS architecture was originally conceived to provide autonomous navigation for vehicles, but its applicability is not specific to that problem domain because it captures fundamental aspects of intelligent agents. The discrete character of agents' interactions with their surroundings, embodied in the control layer, is a general feature of A.I. problems. Also, the general problems of action ordering and prioritization, along with failure response, are described well by the architecture's sequencing layer.

Furthermore, the deliberative layer's internals are not specified by the architecture. The deliberative layer can be as simple or complex as needed to achieve adequate performance in the problem domain; it is the agent's brain, and it can be tailored to solve a broad range of problems. It is in the deliberative layer that the agent "comes to life". Although there is coupling to the sequencing layer, the deliberative layer is fully separated in ATLANTIS from the control layer. The control layer code can therefore be reused for multiple implementations of an agent's deliberative layer. Once the capabilities of the entities in the nation-state model are defined, their level of sophistication can be developed incrementally. In future work, we will compare the

performance of different implementations of the three-layer framework described in ATLANTIS, ranging from simple rule-based systems to representations of rational and even non-rational thought.

References

1. Beyeler, W.E., et al.: A General Model of Resource Production and Exchange in Systems of Interdependent Specialists. Sandia Report SAND2011-8887 (2011)
2. Schildt, H.: Java: The Complete Reference, 8th edn. McGraw-Hill, New York (2011)
3. Gat, E.: Reliable Goal-Directed Reactive Control for Real-World Autonomous Mobile Robots, Ph. D. thesis, Virginia Tech, Blacksburg, Virginia (1991)
4. Gat, E.: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In: Proc. of the 10th AAAI Conf., pp. 809–815 (1992)
5. Jones, T.M.: Artificial Intelligence: A Systems Approach. Jones and Bartlett (2009)
6. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson Education (2010)
7. Holub, A.: Holub on Patterns: Learning Design Patterns by Looking at Code. APress (2004)
8. Gat, E.: On Three-Layer Architectures, A.I. and Mobile Robots. AIII Press (1998)