# Providing a Broad Spectrum of Agents in Spatially-Explicit Simulation Models

**John Anderson**
**Department of Computer Science**
**University of Manitoba, Winnipeg, Canada R3T 2N2**

## *1. Intelligent Agents and Individual-Based Modelling*

Research in natural resource management may be characterized as a search for an understanding of patterns and processes relating to a particular resource. Modelling is a crucial tool to these efforts: resource scientists use such models to help them conceptualize, understand, test, predict, or assess various aspects of the resource being studied [Walker and Cuff, 1988]. One central function, however, underlies all of these uses: a model simulates the way in which a real system would behave under conditions of interest to the user, and illustrates changes over time. Such a model may be used to determine the consequences of particular situations, leaving judgement of the attractiveness of those consequences to the user [Friedland, 1977]. Particularly in the case of complex ecosystems, such a model may also serve to clarify interactions and contribute to a deeper understanding of ecological phenomena [Gigon, 1987].

In recent years, computer-based models have become the most significant tool of resource managers, for two reasons. First, any model must accurately portray the real system it represents if research based on the model is to have any reliability. The use of computer technology has greatly increased the extent and the detail to which ecosystems can be modelled, and thus the accuracy of these models. The other reason for the extensive use of computer models is the flexibility that the computer as a tool brings to the modelling process. Many ecosystems are poorly understood, and complex models for such poorly understood systems are almost never completed. Rather, modelling such a system is an iterative process, with a partial understanding generating new hypotheses, which in turn generate changes to the model based on further research [O'Neill, 1975]. Computer technology brings flexibility and ease of modification to the modelling process, naturally supporting this iterative development. In addition, as the alternatives available in resolving resource management problems become increasingly expensive, and the resources themselves become increasingly scarce and valuable, such models become vital tools not only in the direct management of resources, but in the control of expenses associated with resource management as well.

Despite the widespread use of computer technology in modelling and the added complexity and realism that this technology brings, there is still a large gap between the models made by researchers and those used by natural resource managers. Many of the models created for specific research projects are not practically useful to natural resource managers, often because they deal only with simplified aspects of particular problems. In recent years there has been a growing realization that resource management science needs to tackle more significant, directly applicable problems, and ensure that the models used in such research capture the complexity of real problems in a manner which is useful to natural resource managers.

The desire to fill this gap, and to in general provide more powerful computer modelling tools capable of significant real-world simulations, has led to important developments in modern computer modelling. Two of the most significant form the foundation of this Chapter: individual-based modelling and the inclusion of intelligent agents as inhabitants of simulation models.

Individual-based modelling is a computational approach to modelling a system through the interaction of atomic models of each individual inhabiting the system. Scientifically, individual-based models provide many advances over more traditional ecosystem models. They do away with the assumption that there is some mean individual that can adequately represent each and every individual in a large population, as well as the assumption that significant interactions take place evenly across populations [Judson, 1994; Kester, 1996; DeAngelis et al., 1996]. Moreover, most individual-based models are spatially explicit, allowing interaction between individuals to occur over a wide range of space (and analogously, doing away with the assumption that significant interactions take place evenly across space). As opposed to a simple mean, individual-based models have the ability to represent the biological, physiological, and behavioural distinctions seen in individuals in the real world. Moreover, because the individual is the atomic unit, the simulation naturally takes localized interactions into account. A model of higher-level entities (communities, populations) thus emerges from the dynamics of individual interactions, in the same manner as the high-level phenomena we observe (and in which we also participate) in the real world.

There are many approaches to individual-based modelling, from creating individuals as cellular automata with simple transition and interaction rules ([Slothower et al., 1996]), to modelling individuals with broader perceptual/effectory abilities (e.g. Gecko [Booth, 1996] and the earlier SWARM system [Hiebler, 1994] on which it is based). However, computing resources and associated restraints on the complexity of individual models are still the largest limiting factors in individual-based models, along with a lack of suitable tools for developing such models. Many individual-based ecosystem models manage large numbers of individuals (e.g. [Wilson et al., 1993; Schmitz and Booth, 1997]) while maintaining only a very small amount of information on each individual or supporting a fairly limited variation on individual physiology and behaviour. The need to maintain more than a few parameters of interest for each individual is also recognized as one of the most significant factors in deciding whether individual-based modelling is feasible [Judson, 1994].

The need for intelligent agents in natural resource management simulations arises in part directly out of the individual-based approach. Since the behaviour of an individual-based model emerges from the individual behaviours and interactions of its inhabitants, the accuracy and ultimate utility of the model itself can be no better than the accuracy of its components. A more comprehensive, flexible, and ultimately more accurate model is obtained by modelling the intelligence inherent in individual inhabitants via their implementation as intelligent computational agents.

While this motivation for including intelligent agents as part of an ecosystem model is readily apparent, accuracy in modelling is really a goal underlying a host of additional concerns. Foremost among these is the growing recognition that the major source of change in many ecosystems (and indeed the major reason for studying many ecosystems at all) is the direct and

indirect impact of human activity within the system. In order to reflect this, an ecosystem model must have the ability to directly include changes and disturbances induced by humans as a part of the model itself. While this statement has been recognized historically (Spofford [1975] for example argues that this is absolutely essential for any ecosystem model to be useful in a management context), there has been little hope of including models of human behaviour of any realistic variety until the relatively recent development of intelligent agent technology. To date, there has still been very little work done in this context (but see [Deadman and Gimblett, 1994, Gimblett et al., 1996]).

Similar arguments can also be made for non-human agents. Animals in general respond to their environment with some degree of intelligence: the representations and processes that form these creatures' decision-making capabilities are simply not as extensively evolved as our own. The need to model intelligent information processes in non-humans will of course vary widely from model to model. In an attempt to model the entire population of mountain gorillas in an individual-based manner [Scahill, 1996], for example, it will be much more significant than say, an individual-based model of a fish population. While it has been demonstrated that the ongoing behaviour of extremely simple animals (generally well below the latter case) can be modelled using extensive stimulus-response systems (e.g. [Beer, 1990]), it is also generally accepted that these have bounds, and that more sophisticated methodologies are necessary to model more advanced behaviours. Even in the latter case, techniques used in simple intelligent agents are applicable: rule-based systems, for example, have been used to model the foraging habits of foxes in a very limited manner [Carter, 1997], and similar examples of using low-level AI techniques to construct simple behavioural models can be readily found (e.g. [Folse et al., 1990]).

As we have seen in this section, spatially explicit individual-based models incorporating intelligent agents do indeed represent a potentially powerful solution to many problems. This application represents a significant area of overlap between the fields of Artificial Intelligence and Ecosystem Management; much more in fact than simply an application of the former in the domain of the latter. The general development process involved in the creation of intelligent agents for most any domain generally relies heavily on software-based testing [Etzioni, 1993]. Software environments are created to examine and test agent principles and components, and indeed, the completed agent may itself function entirely within a software development. As such, the development of intelligent agents for most other problem-solving environments essentially incorporates the development of individual-based models on a scale that is small from the perspective of the number of agents involved. Compared to most ecologically-related individual-based models however, the environment itself is highly sophisticated, as the focus of the model is generally upon the problem-solving efforts of the agent itself (and the associated low-level interactions with the domain) as opposed to emerging characteristics of the environment as a whole. What this means is that AI can benefit greatly from the many other modelling advances and general rigour of ecological modelling, and ecological modelling incorporating intelligent agents can benefit from many analogous problems that AI has already had to deal with.

To begin reaping these benefits, however, we must move from the realm of theory to that of physical embodiment. Thus we must turn to examining the characteristics of these models from the point of view of a tool for building them, with the ultimate goal being the construction of

such a tool. Section 2 outlines these needs, and the remainder of this Chapter describes Gensim, a tool that attempts to meet many of these needs though an integrated facility for defining and working with a broad range of intelligent agents.

Before moving on however, two important clarifications regarding the nature of agent technology should be made. First, the near-ubiquitous nature of the term itself. Agents are to contemporary software technology what objects were until comparatively recently. The term "agent technology" has become a buzz-word that is associated with many expensive software packages, and touted as the future solution to many problems, in much the same way that the term "object technology" used to be employed. As with any contemporary darling of the computer software industry, the use of the term should be viewed with some suspicion, and rightly so. We hear of programs associated with web sites, for example, that are relatively simple search programs one day and "agents" the next, with no changes to the underlying software whatsoever. Similarly, many "agent-based" help facilities are simply heavily anthropomorphized keyword indices.

That is not at all to say that all agents are such vapourware: indeed, it is to some degree in the nature of the term for such misuse to occur. At its simplest, the idea an agent is a systems metaphor - a perspective from which to view and study a particular system. An agent is simply a program that perceives its environment and acts upon it [Russell and Norvig, 1995]. While the agent approach has been an incredibly useful tool for bringing areas together and encouraging a wider viewpoint of particular problems, the generality of the term, like that of object technology before it, has left it open to abuse. Russell and Norvig [1995] point out that under this definition, the daylight-savings-time adjustment routine that waits in the background on personal computers and alters the clock on the correct days of the year is an agent. Indeed, so are many other simple machines. A true intelligent agent, however, is exactly that: one that adopts the agent perspective in its design (which is a natural one for anything inhabiting an artificial world) and displays intelligence to some degree in its behaviour. The nature of the term intelligence is also a vague one, but several helpful definitions, from avoiding unnecessary search [Newell and Simon, 1972] to adopting a rational stance [Russell and Norvig, 1995] can easily be employed to ensure some substance exists where the term is implied.

It should also be noted that the use of intelligent agents within a natural resource management setting is by no means limited to intelligent inhabitants in individual based models as discussed here. The interpretation of such models, for example, is also an important part of overall resource management. The need has long been recognized for the inclusion of not only the concise, measurable, decision-theoretic aspects of traditional computer simulations, but also the informal judgement and intuition of the resource manager in the interpretation of such models [Walker and Cuff, 1988; McArthur, 1980]. Agent-based techniques can be used in conjunction with more standard expert systems techniques to achieve these goals. Other diverse applications such as data mining and information filtering are also good applications of this technology.

## *2. Modelling Tools and Breadth*

In order to allow functioning intelligent agents within an ecosystem model, two things are required. First, a model of the behaviour of the desired agent (that is, an intelligent agent design) must be constructed, and some underlying facility for the implementation of that agent's

decision-making abilities must be available. As discussed earlier, these abilities will differ greatly with the domain and the expectations of the sophistication of the agent. This translates into a great deal of variation in the underlying implementation of the agent model. From the perspective of a physical implementation, such models may range from architecturally simple, purely reactive approaches for extremely primitive agents (e.g. [Agre and Chapman, 1987]) to sophisticated systems integrating reactive and deliberative reasoning (e.g. [Anderson, 1995]).

Secondly, and more significantly from the point of view of general natural resource management, the modelling tool itself must provide for the inclusion of intelligent agents. This requires great sophistication in comparison to traditional modelling tools and demands many features of a modelling tool that are not generally associated with ecosystem modelling. Not only must an environment be modelled in a spatially explicit manner, for example, but a realistic perspective of this model must be given to each agent in order that it may make decisions in a manner similar to that of its physical counterpart. That is, the simulated world functions as both an environment unto itself, and a virtual reality to the agents that inhabit it. Likewise, the competing effects of many different agents must be carefully managed, all under the constraints of reasonable timing and reasonable consumption of computational resources. Such tools must also bridge the practicality gap described in the previous section: they must be useful for realistic problem-solving situations, as well for simple research models.

While it is certainly possible to construct completely specialized environments from scratch for each new project (though this occurs in both ecosystem management and AI with unfortunately high regularity), we would ideally like a tool that is flexible enough to support a wide range of agents and environments. The reason for this is not only the desire to avoid reinventing the wheel. Control and verifiability aspects, for example, are involved as well: a tool allows us to ensure that implementations under comparison have identical constraints placed upon them, and helps to ensure scientific validity in the results obtained. More importantly, however, a flexible tool is required because of the complexity of the agents and environments themselves. Complex software systems can neither be completely understood nor completely designed in advance. This will sometimes necessitate changes in design approach in mid-stream, and if our original tool does not support such a change, the tool must be abandoned and a new one chosen. In the worst case (designing specialized environments from scratch for each new project), we have to completely re-build the entire environment and agent *de novo* with each small design change.

This is an extremely common problem in AI, which is arguably the branch of computer science whose systems are by necessity the most poorly understood at the outset of the design process. As such, many specialized design techniques have been developed in this field to alleviate the complexity associated with the applications being constructed, most notably prototyping techniques for commercial expert systems. Like the problems associated with a changing ecosystem model or agent design, changes occur in the model of how an expert performs in his or her domain. These in turn necessitate either small changes, if a tool is flexible enough, or a complete abandonment of a given tool and reimplementation in a tool more suitable to the revised model. The latter is termed a *paradigm shift* [Hayes-Roth et al., 1983; Waterman, 1986], and is analogous to what would be necessary in an ecosystem model if we do not have a tool that can bend and flex with the changing needs of the modeller as a more accurate picture of the domain emerges. In ecosystem modelling (and in intelligent agent research apart from expert systems as well) paradigm shifts are exacerbated by the experimental nature of the domain. In

intelligent agent research, we are often experimenting with different categories of agents in the same domain, or the effects of varying domain characteristics on a particular agent design, putting even more emphasis on the flexibility required of a modelling tool. Similar ongoing alterations in agent and environmental characteristics will also affect ecological models.

Out of this general desire for flexibility, we can construct several more specific requirements of a modelling tool:

- Interaction between agents must be supported, to as sophisticated a degree as possible.

- The nature or implementation of intelligent decision-making in individuals should not be assumed. That is, the simulator should support an interface between agents and their environment, and a mechanism for agents to make decisions in time synch with the rest of the environment, rather than putting constraints on specific agent designs or methodologies.

- The nature of the domain should also be as assumption-free as possible. Many simulators for intelligent agents, for example provide only for grid-based domains (e.g. [Pollack and Ringuette, 1990] and later work) for example. The environment must be able to include a wide range of objects, including agents of limited intelligence as discussed previously, and also the inclusion of non-living objects that can directly affect the environment and the other organisms inhabiting it.

- A simple but realistic model of perception must be provided: agents cannot simply react to other species by virtue of the simulator's knowledge that they occupy the same location in an ecosystem model, for example. Agents must be able to look and see around themselves, and gather information as their perceptual abilities would normally allow them. Similarly, the effectory capabilities of agents should be realistic and individually limited in scope. Agents in an individual-based model should not be forced to implement the same actions as other agents in precisely the same ways.

- Last but certainly not least, the simulator must as much as possible provide means to lessen the computational load required to support large numbers of agents.

These points are by no means independent: for example, supporting varying effectory abilities for each intelligent agent is the most realistic approach, but demands that more information be maintained for each particular agent, increasing the computational load.

There is a general theme of *breadth* among these points: one that makes significant demands on the underlying structure of our simulation tool. The most obvious sense of this breadth is in the broad agents such a tool must be expected to support. In AI, the term "broad agents" is most often used to refer to agents that can be employed for several purposes or that combine several aspects of intelligence, such as reasoning and communication abilities [Bates et. al., 1992]. That is, the term is used to refer to the breadth of the capabilities of the agent itself. Here, on the other hand, we refer to the huge range of possible agents a single tool may be asked to support. This includes not only between simulation applications, where it might be feasible to switch simulator modules geared toward specific types of agents, but also within a simulation, where we may

have human-level intelligent agents[1] as well as a range of animal agents with varying degrees of intelligence all inhabiting and interacting with each other in the same environment cotemporally. This requires that the approach used to support the agents involved in a simulation be capable of dealing an enormous breadth of possible agent models, and have the computational ability to feasibly support many such models (and many instances of each) simultaneously during a simulation. Moreover, the support of individual agents within this spectrum must be done in such a manner as to balance realism and computational tractability.

The fact that agents at one end of sophistication require very different facilities than those at the other, in tandem with an inability to assume the needs of any one particular simulation application (required population of different agent types, granularity of the environment, sophistication of agent interactions, sophistication of timing, etc.) poses a significant challenge for a modelling tool to say the least. Consider just one example: providing a visual interface to an intelligent agent. In an AI simulation we might be interested in the formation-following abilities of agents in a crowded area. In such a simulation, a great richness of vision (such as the ability to pick out colour differences between every single tiny component of every object in the environment) would not only be relatively useless from the point of view of the design of the simulation, but would slow the simulation down tremendously. In a second instance more closely involving ecological modelling, we might be allowing intelligent agents to roam an area, and be interested in the visual contacts they have with one another, or with aspects of the environment that might be less than pleasant. The latter would require, from the same tool, a significant visual acuity and the representation of most concepts in the environment from a visual perspective.

Also significant from the perspective of real-world problems is the tradeoff between the diametrically-opposed metrics of realism and computational tractability themselves. On one hand, the sophistication of resource management problems requires detailed models; on the other hand, we model in part to explicitly avoid representing everything: we place controls on the complexity of the environment and examine (hopefully) those parts that affect the context of interest. Using the same example of a visual interface, we would likely want to skip above the pixel-level of vision, and describe the situation in a more object-level manner.

The many levels of breadth described here act in concert to define the ultimate desireable breadth of our modelling tool itself: what the tool will and won't support, and much more importantly, what kind of effort will be required on the part of the developers to construct models. Tools in general are extensions of our own abilities, designed to enhance those abilities by way of speed, accuracy, or extent. As such, a tool is designed for a specific purpose, and while improvised use of such a tool may also allow it to be used in a variety of scenarios for which it was not initially intended, the effectiveness of that tool wanes as its use diverges from its original purpose. As the adage[2] states, "for every tool, there is a task it fits", and this is certainly as true with software tools as it is for hardware tools. When developing an expert system, for example, we can choose

---

[1] Or really, as close to human-level performance as technology currently permits.

[2] Davis' Law.

between many ready-made system-building tools (see Carnegie-Mellon University's AI Software archive for an extensive list of freely-available tools alone), offering very powerful techniques used for specific applications. Each of these tools differs in features and more importantly, in the degree to which the tool adequately matches a developer's task. Similarly, we can use powerful predefined modelling tools to construct an individual-based model (though the range of tools available at this point in time is comparatively much smaller). However, we will spend considerable time fighting the aspects of the tool that don't closely fit the representational and/or computational needs of the desired model.

Alternatively, a programming language can be considered to be a system-building tool, albeit at a very low-level. It is certainly more flexible than the tools described above: rather than attempting to fit square pegs into round holes, we can build any computational structure we want. The price to be paid for this is great - this is essentially the case we started out with when justifying the need for flexibility in a modelling tool, and brings us all the difficulties already discussed above. However, these two basic alternatives are a common tradeoff, not only in individual-based modelling and expert systems, but over and over in AI in general. The tradeoff between weak, universal methods and strong, specific methods is well known and discussion on this topic be found in any basic AI textbook (e.g. [Rich and Knight, 1991]).

The important thing to recognize is that these two alternatives form two ends of a very wide spectrum, as shown in Figure 1. While tools with completely pre-defined components leave little flexibility and thus are difficult to fit to domains that don't perfectly match the tools' original intent, tools that are completely programmable can be fit to any domain but lack the rapid alteration capabilities required for flexible modelling. While neither of these on its own is a good choice, the optimum lies somewhere between these two poles. Put another way, we'd like a tool that has many predefined components, so that many different types of agents and environments can be built relatively quickly, with programmable aspects so that the tool can better fit our needs. Thus, there is a range between a domain-specific tool and a programming language that we would like our overall simulation tool to cover. Translating back to AI terminology, there is a desire to have the advantages of both weak and strong methods.

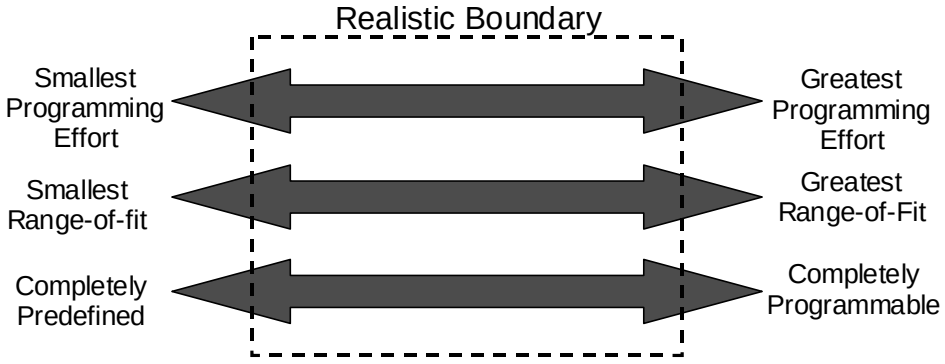


Figure 1. Spectrum of choices in a Modelling Tool

Our place on the spectrum of programmability directly translates to the amount of effort it will take to construct a model directly using the tool, as well as the likelihood we will be able to successfully construct the model using the tool. By "directly", we mean the amount of effort

required that is directed completely toward constructing the model, as opposed to tweaking elements of the tool to try to get it do what we want. To most model-builders, who are not primarily computer scientists, programming effort is also important in and of itself. Despite the desire for programmability to achieve the greatest range-of-fit, it is also desirable to minimize the overall programming effort - or at least make the necessary programming as straight-forward (intuitive, and more important, standardized in methodologies across tasks) as possible. Thus, on these perspectives there is also a range we would like our tool to cover.

So the question becomes, why not develop a tool that will support the entire spectrum across all these interconnected facets? The answer lies in what current technology can hope to achieve. While having a single tool be useful across the entire spectrum depicted in Figure 1 is certainly the ultimate goal, being able to develop such a tool would require solving most of the open problems in knowledge representation and reasoning in artificial intelligence, as well many in programming languages and indeed, in simulation modelling itself. A much more reasonable short-term goal is to develop tools that will cover as wide a range of this breadth as possible, in a practical manner.

The remainder of this chapter describes Gensim: a tool for spatially explicit simulation models incorporating intelligent agents that attempts to encompass much of this breadth. Gensim divides the problem of supporting intelligent agents into the two parts discussed at the beginning of this section: providing support for the integration of agents and their specific needs within a spatially-explicit simulation tool, and providing a means to model the computational abilities (decision-making, perception) of intelligent agents. Both components attempt to cover as much breadth both in terms of the variety and sophistication of agents supported, the ease of their definition, and the range of the spectrum depicted in Figure 1. The two major components of the Gensim approach and the connections between them are described in the following sections.

## *3. Gensim and Low-Level Support for Intelligent Agents*

Gensim is a LISP-based object-oriented simulation system designed explicitly around the features necessary to support intelligent agents in spatially explicit simulation environments. The system operates in a uniprocessor environment (though one of our future goals is to use the system as a basis for larger distributed simulations) and supports multiple intelligent agents, each of which may consist of multiple timeshared processes. The system has a concise interface between agents and the rest of the environment, clearly defines all restrictions and assumptions regarding the agent-simulator relationship (e.g. action timing, perception) and provides the ability to control many aspects of the simulation itself. Because the system was originally intended for developing and working with intelligent agent research [Anderson, 1995], breadth in agent support was one of the foremost considerations of the system itself. Gensim thus incorporates many modular design aspects, making it relatively straightforward both to modify the features provided with the system and to add particular features required to support a given agent or domain structure.

Rather than simply providing a flexible domain or set of domains, as is done in many AI-based simulators for intelligent agents (e.g. [Engleson and Bertani, 1992]), Gensim provides flexibility in the simulation process itself. That is, rather than providing extensive domain parameters, Gensim provides the features necessary for users to define their own domain and their interfaces

with intelligent agents. The system also incorporates facilities for constructing intelligent agents, which will be described in the next section.

A high-level overview of Gensim is illustrated in Figure 2. The simulator itself manages the environment, represented as a collection of objects. This environment is spatially explicit and *objective*: it is the universal set from which all intelligent agents' perspectives are defined. The simulator also possesses a collection of procedural knowledge describing the actions that agents can perform on these objects, as well as the physical events that can occur to these objects outside of the influence of any agent.

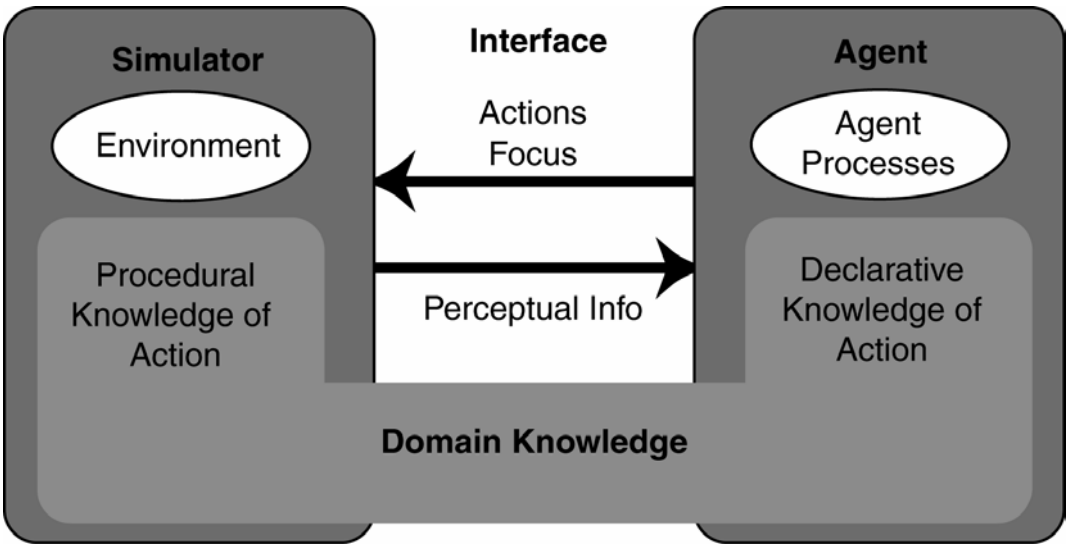


Figure 2. Conceptual View of Gensim

A collection of agents is also defined, each with its own view of the environment based on its sensing ability and memory. The reasoning abilities of an agent are implemented as a set of time-shared processes, which collectively allow the agent to perceive the environment around itself and act on the basis of those perceptions. Agents gather information passively and actively from the environment around them (i.e. from the simulator), and are given limited time periods in which to process that information and commit to actions. After each agent is given a time slice, a representation for the activities of each agent over that shared time slice is obtained and the simulator then manifests the effects of each agents' actions, both short-term and long-term, on the simulated world. These manifestations include both the effects of (from the agents' points of view) intended and unintended interactions between agent activities.

Agents are viewed as "black boxes" by the simulator, in that Gensim neither knows nor cares about their internals nor how the agents arrive at their decisions for action. As indicated in Figure 2, some commonality in domain knowledge is required in order for agents to interact with the simulator. For example, the actions that the agent can select from (represented largely in declarative form) must be identifiable by the simulator, which then uses its own (largely procedural) knowledge of action to physically modify the environment appropriately. Agents also inform the simulator of their interests in objects in the environment (their *focus*) in order to provide appropriate sensory feedback. However, emphasis is on making agents' knowledge

distinct from that of the simulator, and the illustration in Figure 2 should not be interpreted as implying that agents somehow must physically share the simulator's knowledge.

The basic design goal of Gensim was to support as wide a variety of agent and domain designs as possible. To this end, Gensim was designed to keep agent and simulator knowledge as separate as possible, and thus limit the knowledge each must have of the other. A complete environment is defined for the simulator, and agents are expected to possess their own knowledge of the objects in the domain and how they operate. This is more complex than simply allowing an agent to share the same internal objects manipulated by the simulator, but allows much more flexibility in implementing complex domains. In particular, it allows agents' perspectives to differ, and limits agents' knowledge to that information the designer of the domain wishes them to possess. It also forms an experimental control on the developer: as one has to explicitly cause agents to share global knowledge, one is less likely to take this route with the naïve assumption it will make no difference to the end result of simulation. On the other hand, there are indeed times when one does legitimately want agents to explicitly share information, and the ability to do so is provided within the simulator. This makes it possible to have agents with varying degrees of autonomy, allowing them to share knowledge of action, knowledge of the world, and even perceptual abilities [Anderson and Evans, 1993]. It also allows us to declare information and abilities common to classes of agents separately from individuals within those classes. From an individual-based perspective, this in turn allows us to support large numbers of individuals without the work of redefining all the knowledge and abilities of each redundantly.

The simulator manages change through a global queue of events (representing future changes the simulator must manifest) to which all agents contribute. An event in Gensim is a packet of change for a specific period of time (which may be initiated by an intelligent agent, some other object in the environment, or randomly in the environment itself). The system is clocked, and the length of time represented by an event will always tie directly to the clock cycle length. Since the clock cycle length can change, allowing for various levels of granularity desired in a simulation, some occurrence in the domain will generally consist of a series of events as defined here, with the number in that series varying by the length of the clock cycle. Actions of intelligent agents and other sources of change are defined as event generators: any sequence of events over time is represented by an event generator generating an event, which in turn invokes the same or another event generator to continue the series. Thus a movement event across distance involves the initiation of movement by an event generator, with the actual processing of that event by the simulator resulting in the invocation of another event generator to continue the movement across the next time cycle. This brings about two very important results from the point of view of efficiency and realism. First, the length of the simulator's event queue is shorter, in that an ongoing event broken into an event series still has only one element of that series in the queue at once: each element generates the next when it is processed. The latter point also allows realistic interaction of events in a computationally feasible manner: each event can have dynamic effects based on those in the queue at the same time, and once again, the shorter queue means that such interactions can be more efficiently managed. It also means that interactions can be handled efficiently at a finer-grained level than is possible if events were represented using more temporally extensive structures.

This approach also allows great flexibility in timing, in that if the time cycle length in a simulator is shortened, each event simply represents a correspondingly temporally smaller packet of

change. For example, in the movement event above, if the time cycle were shortened, each event simply represents a smaller distance travelled in that time. It does require that the definition event generators take time length into account, but in most events this is a fairly natural thing to do (and indeed, what one would be desireable for a realistic simulation regardless). The ability to support a variable-length system time cycle in turn brings another important aspect of flexibility to the process of individual-based modelling. The basic cycle length is irrelevant from the point of view of time within the simulation, in that shorter time cycles will simply result in a slower execution of the simulator itself. However, large numbers of sophisticated agents combined with short time cycles may slow the simulator down enough that the overall time of a simulation run would become impractically long. Allowing variance in time cycles supported directly in the representation for change in the simulator itself allows us to tune this level for the number of agents we have: discretizing time within the domain differently, but speeding the overall simulation. For a more fine-grained level of detail, short time cycles can still be used (to the degree one is tolerant of a slower simulation) without changing the definition of the environment itself.

Further details and analysis of this approach to timing may be found in [Anderson and Evans, 1995]. This model of timing is extremely flexible and more than adequate for most ecosystem modelling domains (which tend not to operate on the low-level of temporal granularity that AI simulations require). It certainly improves upon the many simple simulators that manage time at the level of whole actions rather than a resource unto itself (e.g. [Agre and Horswill, 1992]). It can be criticized, however, on one significant point: since the processes of an agent (and indeed, the agents themselves) are serially timeshared, they cannot simulate some of the interactions possible in truly parallel processes. For example, an agent might consist of two processes: one to recognize objects in which it has an interest, and another to decide what to do based on the objects it has seen. When these run in parallel, the acting process would process input from the recognition process as it became available. This is a much more complex interaction than that possible under timesharing in a uniprocessor system, where the recognition process must recognize all the objects it can in a given interval and then pass the entire collection to the acting process at once. This is a legitimate criticism, but like the accuracy of a sensory interval, where a short enough interval becomes equivalent to the agent to continuous data flow, the effects of this depend entirely on the interval size. Like other applications of timesharing, the smaller the time-slices involved, the more transparent the timesharing will appear.

The timing model Gensim is designed around is at a higher level representative of the modelling approach adopted by the system in general. Referring back to Figure 1, this approach attempts to take in most of what would fall within the realistic boundaries of expectations of a simulation tool. The model is flexible, straightforward to modify, and encompasses most of what would be needed for the majority of modelling scenarios. Situations such as that described above can be identified that do not fit the model well. However, including them in Gensim's model of timing would result in a system that is only marginally more capable than it would be otherwise, at a cost of a significant decrease in ease-of-use and short-term flexibility in general.

A Gensim environment is managed in an object-oriented fashion. When an event is processed, for example, messages are sent to the objects involved, which in turn evoke changes to the environment, and possibly new events. Say, for example, a particular agent performs a `throw` action on a particular object that is part of the environment. That decision would be based on the

agent's declarative knowledge of the object it's throwing, what its throw action does, and the other overall knowledge the agent has that motivates it to perform this action. That fact is communicated to the simulator by the agent during its decision-making time-slice. When the simulator process runs, the initial changes are made by sending a `throw` message to the object being thrown (the simulator's version of this object), which contains knowledge dictating the objects' behaviour (e.g. how fast it would travel, whether the trajectory could change spontaneously). A series of travelling events for the object would ensure over time, possibly interacting with other events in the queue at the same time. For example, if two objects were travelling and entered the same physical space, a `hit` event could be generated for each. Once again, the simulator could send this message to each object, resulting in the manifestation of change in the environment based on the physics defined. A more detailed description of this overall process may be found in [Anderson and Evans, 1995].

In addition to performing actions, agents also interact with the simulated world through perception. Perception is the most difficult design aspect of any simulator for intelligent agents: on one hand, it is a crucial aspect of virtually any domain, while on the other, simplification of perception is in AI research one of the primary motivators for using a simulator, and the greater the simplifying assumptions, the smaller the range of agents we can accurately support.

The perception abilities of intelligent agents are much more active processes than might first be assumed. We do not simply see whatever our eyes happen to be directed toward, for example: we pick and choose from the image, discerning what interests us. What we have already seen also alters further perception, helping to define a focus for perceptual efforts. In more computational terms, a set of *anticipatory schema* (domain-specific knowledge indicating the sensory information in which the agent has an interest) acts as a filter for the vast amount of knowledge available from the world. This knowledge directs the agent to explore the world, looking for given pieces of information. This exploration directs the agent to specific information (objects) in the environment, which then modifies the anticipations the agent has for future sensations [Neisser, 1976]. Within this cycle, one of the basic problems of implementation is that an agent must be allowed to specify its interest in given aspects of the environment, but must also be given access to information independent of those interests. Perception must exist to confirm the agent's expectations of the world, but also must provide the agent with new information (not necessarily what it expects to see or is directed toward) in order to form the expectations of the world that guide its sensory abilities.

Gensim attempts to follow this model as closely as is practical. However, some aspects of perception are simplified, in order to simplify the simulator itself and to conform to one of the basic design goals of Gensim: a simple interface between agents and the simulated environment. One of the major simplifications concerns the internal processing within this cycle. Rather than beginning with low-level vision, Gensim operates at the object level. That is, an agent senses a combination of complete objects and specific sensory aspects of those objects (e.g. a ball, or the fact that it is red or round), rather than examining the individual edges and features that make up an image of the object itself. Agents perceive objects through a symbolic description of the visible aspects of the object. They make sensory requests by describing the direction they want to look in, or attributes of the objects they wish to look at. A bandwidth can be set for sensory information in each agent, limiting the number of objects the agent can focus on and the amount of information that can be received at once. To go lower than the object-attribute level would be

outside the realm of most ecological modelling work, as well as most work in AI from the agent perspective, and would immensely overcomplicate the process of defining and working with a domain from the modeller's point of view.

The ability of an agent to express a focus for its sensory abilities is provided in Gensim through a sensory request mechanism. This mechanism allows an agent to express interest in certain aspects of its environment, and the methods by which Gensim fulfils sensory requests allows the agent to receive limited sensory information not only about its focus of interest, but also about objects outside that focus. An agent makes a sensory request for some particular set of information, and the simulator records this request. After all agent processes have been run on a particular cycle, the simulator updates the environment according to the actions and events that have occurred during the interval, and then prepares sensory information based on each agent's requests. Gensim has defined sensory actions that allow an agent to look in a given direction or angle in the environment, and to look for objects matching particular templates.

Like physical actions, sensory requests form the interface between the agent and the simulator: they are the agreed-upon vocabulary the agent will use to interact with its environment. As such, like actions, part of the definition must lie within the agent (what it thinks the action will do), and part within the environment (the actual effects of the agent performing that action). Thus when defining the domain, the physical results of a sensory request must be defined for an agent. That is, what kind of objects or attributes an agent will be biased to see over others: allowing larger objects to be seen first, for example, or decreasing the accuracy of sensing (the likelihood that particular attributes are perceived) as distance increases. This sounds time-consuming, but is in fact quite simple compared to many other aspects of domain definition. All perception is a filtration process, here selecting a bandwidth-bound group of object attributes from the simulator's knowledge base. A simple set of rules can easily bias an agent toward one type of information, and many of these rules are similar from one perceptual action to another and can simply be duplicated (the angle representing an agent's perceptual boundaries, for example). Moreover, once again in keeping with the idea of supporting a broad range of simulations, this level of detail can simply be avoided entirely by increasing an agent's bandwidth and not biasing the agent away from receiving any particular attributes of an object over others. That is, limited, biased perspective can be easily avoided for the many simulation environments in the spectrum of Figure 1 in which it is deemed unnecessary.

Defining a domain in Gensim involves constructing the objective environment that the agents inhabit, the knowledge base for each agent, and the behavioural mechanisms (the agent design) of each agent. Once again, Gensim attempts to take a middle of the road approach to the spectrum of inflexible pre-defined elements vs. complete programmability as shown in Figure 1. Many elements of an environment are defined using simple macros, making construction of a simulation environment fairly straightforward. For example, a `defaction` macro defines an action, names it, and defines a procedure describing how the action manipulates the object it uses. Similar macros are used to define potential events, as well as to define agents inhabiting the environment. One still must program the physical effects of the action; however most aspects of this are common to many actions (transferring possession, creating an object, destroying an object, etc.) and have library elements that can be used to eliminate at least some of the low-level programming involved.

While macros define the objects that exist in any particular domain, other aspects of a domain are set using parameters. For example, a single parameter specifies the number of dimensions in a domain. The user can then define a function that maps the domain out in a regular mathematical pattern such as a grid (most common mappings for simple structures such as grid-based domains are provided), or as a collection of symbolically named locations. The latter allows domains where areas cannot be mapped into a regular grid structure: significant locations can be grouped and designated with a symbol, allowing an irregularly structured area as a single location. This is similar in functionality to the use of GIS information by Folse et al. [1990]: related areas (e.g. a single location with similar vegetation or soil structure) are grouped into a single area that is treated as a unit for simulation purposes.

As has been stated throughout this section, the major features of Gensim are all designed toward making the simulator available relatively easily to a very broad range of environments. While lines have been drawn in terms of facilities not available to intelligent agents associated with the simulator (e.g. parallelism, as mentioned above), in general the reasonably clean, extendable interface between simulator and agent allows for an extremely broad range of potential agents. While not directly interfaced with any commercial GIS tool, the general spatial organization underlying a Gensim environment is not unacceptably far-removed from the approach used by such tools.

Gensim also directly supports the management of computational resources in individual-based models in several ways, the most significant of which arise from the system's object-oriented nature. Each intelligent agent in a Gensim environment belongs to a class that describes not only shared attributes, but shared abilities. It is thus possible for many different agents to share perceptual abilities, effectory abilities, and knowledge of the world. Classes are organized hierarchically, and so at any point lower level classes or individuals can redefine knowledge or abilities assumed by higher-level classes. Moreover, the same code may also be used to make decisions for many agents at once. The system thus directly supports a wide variation of individual abilities and knowledge with minimal redundancy and greater conservation of computational resources. Gensim can also vary the time each agent is allowed to make its decisions, giving agents (or classes of agents) intellectual abilities relatively higher than others. Finally, these mechanisms are generic enough to easily support a broad range of agent abilities within a single model.

When outlining the basic requirements for incorporation of intelligent agents in a spatially explicit simulation (section 2), we began with a rough division into two major component groups: support for the design of agents themselves, and support for their realistic interaction in a simulated environment. Throughout this section, we have shown the applicability of many of the features of Gensim to the latter problem. However, considering the breadth of agents desired and the desired range of breadth necessary for the tool itself, the former remains a significant problem.

The problem is certainly not that intelligent agents cannot be employed using only the features of Gensim described thus far. Within any Gensim simulation, an agent can be any collection of LISP functions (simply because the simulator itself is currently implemented in LISP, a holdover from Gensim's origins as a testbed for intelligent agent research). This means that if one so desires, any agent design that can operate by describing actions to the simulator in the manner

that Gensim expects, requesting and accepting perceptual information in the manner described above, and adhering to the Gensim timing model can be programmed for the system.

While several general classes of agents have been developed for use with the Gensim system, forcing modellers to work with the simulator in this fashion is contrary to the breadth desired of the simulator itself. By programming agents individually from scratch, we are back to the potential paradigm shifts described in section 2 at another level: a slight change in an agent's design or purpose requires potentially extensive reimplementation. What is required to deal with this is a framework for agents themselves that provides benefits at the agent implementation level analogous to those the features of Gensim bring to the agent-support level. That is, a framework that supports within a single basic design the ability to define a broad range of agents capturing a variety of degrees of intelligence; the ability to modify those agents quickly; and the ability to alleviate some of the difficulties of declaring and working with large numbers of agents. Such a framework is described in the following section.

## *4. Improvisation and Intelligent Agents*

The approach to agent design used to provide for breadth of agents in Gensim arose out of a study [Anderson, 1995] attempting to explain and duplicate some of the breadth of ability displayed in everyday human behaviour. This work created a model that, while certainly not claiming to implement and display the full spectrum of human behaviour, implements techniques that permit a wide range of adaptable behaviour, in a manner that can be employed to deal with other forms of breadth as well. To see the utility of the approach in supporting a broad range of agent types in an individual-based simulation, we must begin with the origins of the approach in extending traditional approaches used to implement agents.

Consider the range of agents required in a sophisticated individual-based model designed to examine human impact on an ecosystem of interest. We need to model the abilities of humans: the activities consistent with their motivations for the use of the ecosystem, and the mental processes underlying their decisions (including for example, each individual's collective previous experiences in this environment). We might also have to represent a range of lesser creatures, with tendencies to particular behaviours, patterns of interaction with resources within the ecosystem, and to some degree the ability to make decisions on their actions. This ability can range from advanced agents that can employ some of the same intellectual resources as humans (again, such as previous experience within the environment), down to those that can operate only at a purely mechanical, stimulus-response level. Once again, this is what causes the difficulty in supporting intelligent agents within individual-based simulations: there is an enormous variance in features required for these various implementations.

The interesting thing about this breadth is that it is also observable within the course of the activity of a human agent alone. While activities such as working on an assembly line (where each potential problem has been experienced many times and there is a limited number of possible interactions) fit a purely reactive model nicely, as does any activity with which we are completely familiar, and which has fairly distinct boundaries around it (in that it interacts in only a limited fashion with other activities). This is one end of the spectrum of human activity. In virtually any human activity, there will be aspects or components of the activity with enough structure (experience on the part of the agent and physical structure in the environment [Norman,

1988; Hammond and Converse, 1991]) to support a compiled collection of responses. During the course of an activity such as preparing a meal, for example, much is routine in a general sense despite the complexity of the environment: we can immediately recall a routine that has been put together over the course of many previous episodes of behaviour (compiled plan or routine knowledge). Portions of this routine with which we are extremely familiar may be compiled to the point of a complete collection of reactions. Conversely, there will also be a large component of any complex activity that cannot hope to have this kind of structure: anywhere where every possible contingency cannot be anticipated. This includes performing an activity with which we are not *completely* familiar, performing the activity in conjunction with others in the short- or long-term, or performing an activity in a different situation than usual [Anderson, 1995].

In order to apply a routine effectively and flexibly in the face of greater variability than can be completely anticipated, we possess a vast collection of more general knowledge that allows us to integrate alternatives seamlessly with our routine. We divert from our routine when it makes sense to do so, and return to it without anything like the kind of effort known to be required (e.g. to alter a stored symbolic plan. We can also use our routine as a weaker guide in conjunction with background knowledge to cope in a satisficing[3] manner with even greater degrees of variation. For example, one can shop reasonably successfully even when in a hurry and in a strange supermarket; can prepare a meal easily in a friend's kitchen; and can sharpen a pencil without a great deal of intellectual work even if no sharpener is available. We commonly call the methodology behind such efforts *improvisation*.

Improvisation as creating minor to extensive variations on a routine in a satisficing manner in real time occurs in the vast majority of human behaviour, from theatre and music to cooking, driving, and architecture [Jencks and Silver, 1972]. In the realm of intelligent agents, the term has been used previously, most notably Agre's [1988] definition of *continually redeciding what to do*. This use, however, leaves out much of what improvisation as described above embodies, most notably a basis upon which to improvise[4].

This basis is the collection of routines through which we normally accomplish our activities and which are constructed through many episodes of similar activity. During the course of improvisation, this compiled knowledge represents a resource that we rely upon to reduce the intellectual effort that would normally be associated with complex activities, in order to perform in a timely manner. We can and do rely on this resource strongly in cases where the current situation follows our previous experience. In situations where our previous experience differs, we can use our routine as a weaker resource. That is, we can follow it as closely as possible (for economic reasons), improvising to obtain alternative actions where the routine is not appropriate

---

[3] Simon [1982, p. 259] coined this common AI term to describe the behaviour of humans and other organisms; as opposed to the ideal of optimizing behavioural strategies, organisms adapt well enough to *satisfice*, or obtain a "good enough" reward to suit their efforts.

[4] Others, most notably Agre and Horswill [1992] and Hayes-Roth et al. [1994 (and more recent Stanford KSL work)] have previously used the term *improvisation* in a more general (and differing) sense than that defined here. See [Anderson, 1995] for further discussion.

by examining the associated background knowledge to the degree the situation warrants. The more novelties there are in a given situation, the less directly the routine can be used, and the more search is required.

This process of improvising on a routine can also occur when one wants to reason beyond the routine aspects that normally constrain our reasoning: when one wants to do better than one's normal routine, for example, or when one wants to come up with creative new solutions using the resources at hand as opposed to those commonly associated with the activity. Improvisation is a naturally satisficing process, allowing the agent to follow its routine and obtain immediate possibilities for action, or to devote as much time as is available or as the agent deems necessary for the task at hand to reason more deeply about alternatives for action.

The range of performance when employing improvisation correlates directly to the wide variety of agents we require in individual-based models. At its most sophisticated, we can have extensive routines for activity, including rich and diverse background knowledge, allowing the implementation of agents representing human-level adaptability in an environment. Yet in the same way that a human might be given less adequate routines or less of an efficient ability to make use of them (e.g. a very limited response time in decision making, a wide range of tasks requiring attention to be divided, etc.), defining less intelligent agents involves applying simpler routines in a simpler manner. This can be extended directly down to routines consisting only of low-level stimulus-response information for the most primitive agents. The end result is a *single* framework that can be used to define all agents irrespective of their abilities. This is one more example of the "middle-of-the-road" approach to modelling within Gensim: by providing a single framework, we ease the burden of implementation required on the part of the modeller while still maintaining the breadth of models suitable to the tool.

Computationally, improvisation has several requirements: control over the extent to which background information is explored, both with regard to the time spent on any one decision for action and the extent to which such exploration is deemed more valuable than the agent's routine response; dealing with limitations on the amount of background information that can be considered at once, and organizing this information such that the most valuable information can be examined first; the integration of multiple goals; and the use of limited perception to recognize new opportunities in light of the agent's intended activities, to name a few.

Our approach to improvisation relies on representing and reasoning explicitly about these aspects of control using constraints, a form of knowledge representation mechanism based on restrictions. A constraint can represent what is and is not permitted, and to what degree, in particular situations. This naturally encompasses most of the knowledge used by an agent in decision making. Individual pieces of knowledge about the domain (what one has done before, specific warning signs, and so on) drive the agent toward or away from specific choices for action. These kinds of information are also hierarchical: we have "Don't Do X" constraints, which we can employ *prima facie*, but which themselves are based internally on the many more detailed constraints that originally led us to decide (or someone else to teach us) that action X wasn't a very good idea. These background constraints exist but remain largely dormant unless we need to enquire why action X isn't appropriate. This naturally fits into the improvisational model: we'd be performing such inquiries (outside of idle curiosity) when we need to reason more about action X than we would normally care to. That is, when the situation is more

important, where action X has potential interactions that do not usually occur, or when our choice between X and other potential actions is not clear.

Similarly, the means of control of improvisation itself is also naturally constraint-directed. How far to explore, when an answer is good enough, perceptual focus, and so on can all be viewed as explicit constraints that describe the state of the agent's decision-making process and evolve as activity progresses. The process of arriving at a choice for action thus becomes a constraint-guided search process [Fox, 1983], where we consider the constraints on an agent's activity in a satisficing manner, controlled ultimately by other constraints on how far the process can go.

Our approach to improvisation, *constraint-directed improvising agents*, relies on a representation based on constraints combined with specialized internal mechanisms for processing constraints in this manner. The sophistication involved in this may sound like somewhat of an overload for something like a simple stimulus-response agent: however, stimulus-response bounds are also constraints at an atomic level, with very little above them. That is, such agents represent the absolute endpoint of simplicity within this model, and simply do not use much of the power (or require as much of the effort to define) as the model is nearly capable of. The agent model as a whole supports both these simplistic level intelligences, and can also be used to describe the vast majority of human behaviour.

A constraint-directed improvising agent's compiled routine knowledge and background knowledge are incorporated into distributed constraint-based knowledge structures known as *intentions*[5], essentially representing the agent's resources for performing some activity in both routine and non-routine manners. A conceptual view of an intention is depicted in Figure 3. In general, an intention consists partly of the agent's routine knowledge of a particular activity; that is, a general description of how a normative instance of the activity should proceed. This description is made up of the constraints that the activity places on the agent's behaviour (for example, preferences for actions or further intentions, or requirements for the activity). An intention also contains links to the background knowledge out of which the agent's compiled routine has evolved. This knowledge also consists largely of constraints, and represents knowledge behind the preferences and other constraints that make up the agent's routine. This division allows the agent's routine to make suggestions for activity immediately where appropriate, and background information to do the same using a search process whose length will vary depending on how closely associated particular pieces of background knowledge are to the agent's routine. This in turn allows the agent access to immediate responses that are useful in the typical case of the activity, and the ability to search and deliberate as time and the significance of the situation permit.

---

[5] The use of the term *intention* here arises from Boden's [1973] use of the term, to indicate representation of both an agent's goals and an overall means for achieving activity, as opposed to a more basic plan structure. The term is also used in other systems, most notably that of Bratman et al. [1988].

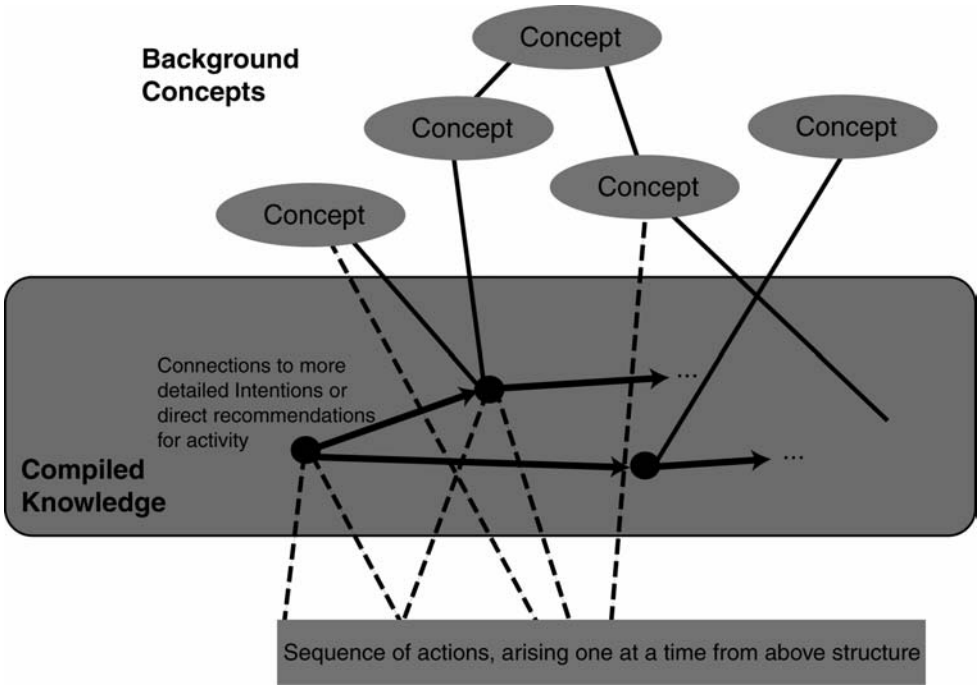


Figure 3. Simple Example of an Intention

Considered in more detail, the compiled knowledge representing the agent's normative routine in Figure 3 is divided into ordered logical units within this compiled knowledge can conceptually be thought of as steps in the routine or plan. These may simply be connections to more specific intentions (just as "travel to location X" might be one step in an overall food-gathering routine, yet a whole intention with many lower level activities in its own right) with constraints supplied to tie the intention to the current context, or direct recommendations for immediate actions. The background concepts are organized in a hierarchy of relationships with one another (taxonomic, geographic, component, etc.) and include information about setting for this behaviour and how it affects the routine, the objects required for its performance and how they contribute and so on. Literally, this information is the collection out of which the routine itself would have been built as part of a learning process (and thus these background concepts have direct connections to the various elements of the compiled portion of the intention). These background concepts are also shared between a great many other intentions an agent might have.

Constraints from both the routine itself and the background knowledge associated with it (to the degree the agent chooses to explore the latter) direct the agent toward individual actions over time. Actions may include adopting further intentions, thus changing the milieu of constraints that form a basis for the agent's moment-by-moment choices for action and the overall context in which any adopted intention is considered. A series of actions thus emerges over time as a result of the initial adoption of an intention in conjunction with others adopted at the same time and independent events that occur in the environment as the intention unfolds. The nature of this process will be explained in much more detail shortly.

As a specific example of an intention, consider a sophisticated human activity such as preparing a common food or drink, such as coffee. A human would have a routine for doing this particular activity, assuming it was indeed a routine activity to that person, and that activity would have

arisen out of (and be connected to) many common concepts: kitchens, coffee, coffeemakers, spoons, cups, water, and so on. If the individual in question was in his or her own home with the familiar utensils it always used, the steps in the compiled routine could be used more or less directly (barring unforeseen circumstances arising in the environment, such as dropping utensils and so on). If the agent was performing this activity in a different locale, however, or with tools of a different nature, the background concepts would become much more important (in proportion to the degree the current situation differed from the agent's routine). Some direct steps would still be applicable at certain times, but new ones would have to be recommended (based on, for example, differences in how coffee-making equipment operated in the new locale).

Intentions such as this particular example are in general much more complex than would usually be required for an ecosystem model. The intentions of human agents are not likely to be this detailed in an ecosystem model (nor in fact will the model itself likely be at this level of detail), and those of lesser agents are likely to be concerned with much more general activities: food gathering, following paths to shelter, hiding behaviour, etc. [Anderson and Evans, 1994]. However, intentions of the complexity of the above example have indeed been implemented [Anderson, 1995; Anderson and Evans, 1996a,b] and applied in real-time environments. In the case of applying this mechanism to ecosystem modelling, it is simply easier to define such structures for simpler situations. The complex example above is illustrated to more adequately illustrate the nature of intentions than a simpler example might, and to show the sophistication this representation structure is capable of.

Constraints in intentions represent regularities in the world around the agent and the influence of those regularities on agent behaviour. Despite the wide range of knowledge being represented, we have found as have others (e.g. [Fox, 1983; Evans et al., 1992]) that only a reasonably small number of distinct types of constraints are needed. Constraints are used in this approach to represent a broad range of concepts, from physical restrictions (*physical* constraints) and relationships between entities (*expectation* constraints, *requirement* constraints, and *temporal* constraints such as the ordering between routine steps depicted in Figure 3) to abstract policies (*behavioural goals* and *preference* constraints), to representing normative responses to particular situations and control of internal agent components (*normative* and *focus* constraints) [Anderson, 1995]. Each of these types has similar components, including a specified active lifetime, activation requirements, and an attachment to some larger knowledge structure. Constraints are organized in a loose hierarchy, abstracting both knowledge and control, and perform different functions depending on the level at which they are defined.

The use of constraints as the primary knowledge representation mechanism directly supports the ability of a improvising agent to perform in real time and to perform in a satisficing manner with the knowledge at its disposal. In an intention such as the one shown in Figure 3, for example, the compiled routine may contain among other things a constraint indicating that the agent should prefer working with a standard drip coffeemaker as opposed to some other tool (an old-style percolator, or the microwave and instant coffee). This constraint expresses a preference that is normally applied in the course of the activity with no exploration as to the reasons behind the preference. When this tool is unavailable or the agent wishes to reason beyond the routine (due to error, knowledge of potential error, or to high-level constraints that affect how an agent performs an activity), the agent can make use of further constraints behind that preference (background knowledge) that describe the role and function of the coffeemaker in the overall

routine. The agent can then use those constraints as a basis for reasoning about alternative ways of performing the activity, to the degree the agent wishes to devote intellectual effort to this. For example, constraints about making will lead the agent to a set of objects with characteristics suitable for this purpose (the tools mentioned previously or further improvised versions of these based on the functionality of these alternatives). Similarly an agent can prefer one path to another, or one form of prey to another, with more detailed knowledge representing those preferences still present to consult if the situation warrants.

Because constraints are modular, constraints external to an intention (e.g. from another, or more global constraints associated with the state in which the agent finds itself) can also have immediate effects on it. The presence of a constraint such as hurrying (brought on by a combination of intentions or some external event) may affect it in certain predictable ways that are part of the routine itself. That is, the presence of a *hurry* constraint from outside the intention may allow certain routine components to become active that would be otherwise ignored. Such a constraint will also affect the agent itself: how much information the agent considers, strategies for deliberation, automatic preference choices, etc.

In any significant domain, there will clearly be a large number of potentially relevant constraints available for an agent's consideration at any point in time[6]. However, the agent's cognitive effort in this approach is for the most part not spent on looking for constraint violations, as in most constraint directed reasoning systems. While we are concerned about violations in some cases (e.g. expectations), here most constraints act positively: their presence compels the agent toward or away from specific courses of reasoning or activity, just as the landscape influences the direction of one's travel. The key to real-time performance is the *selective* processing of constraints in order to make satisficing decisions in the time available. This is done through the multi-level organization of constraints in intentions, in tandem with facilities provided in the agent architecture for limiting the number of constraints considered.

The architecture of a constraint-directed improvising agent is shown in Figure 4. The agent itself is divided into several computational processes (which would be timeshared by Gensim) and two major stores of knowledge. The agent's *long-term memory* contains all the possible intention and conceptual knowledge possessed by the agent. The central role in this architecture, however, is played by the agent's *working memory*, which directly supports the selective recall and processing of constraint knowledge over time. Working memory is of limited size, and represents the amount of information the agent can process in parallel (in our implementation a timeshared simulator is used, and the size of working memory represents the amount of information the agent can process in a time-slice). Any constraint in working memory is thus viewed as having immediate effects on reasoning or behaviour, and relations among concepts in working memory are assumed to be immediately realizable (via direct memory connections). These are not unrealistic assumptions, since the number of concepts or intentions physically

---

[6] Once again, given much simpler intentions than that depicted in Figure 3 this may not be significant, but this point must still be dealt with because of the potential for building very sophisticated agents.

allowed in working memory at the same time can always be set to a small enough limit to make this so.

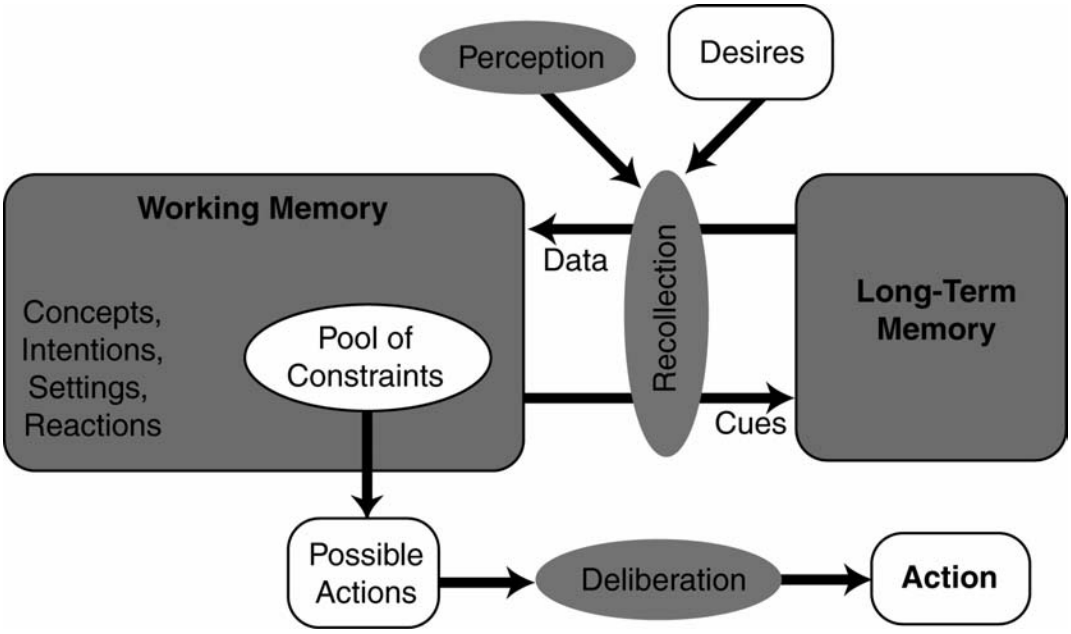


Figure 4. Computational Aspects of Constraint-Directed Improvisation

Items migrate from long-term to working memory through the use of memory triggers. Each object, concept, or intention may have simple trigger conditions associated with it that, when satisfied, allow this item to be brought into working memory. Trigger conditions may be satisfied through perception, through new desires, or through objects already present in working memory. Trigger conditions are only considered when present in working memory, and are brought into working memory when adjacent or connected concepts are present in working memory [Anderson and Evans, 1996a]. This directly supports the ability to gradually move to more detailed background knowledge over time.

At any point, the collection of constraints in the agent's working memory will give rise to a particular set of alternatives for action, which the agent can deliberate over. The more constraints available to working memory, the fewer alternatives need be deliberated upon, and hence deliberation is secondary to working memory in this approach. However, because working memory is very limited, the agent will often have only an incomplete picture of its overall choices and have to decide if further exploration is warranted, and so deliberation is still significant. Deliberation in the this architecture is dealt with using constraints on the amount of cognitive effort that can be directed toward a particular end. This illustrates the most significant aspect of this approach: not only do constraints represent the structure of an agent's activities, they also control the architecture itself.

Individual constraints in working memory influence the agent toward (or away from) some alternative for action through a quantitative or qualitative measure of utility. This measure is composed of an innate estimate of the constraint's importance, modified by the importance of the chain of intentions through which the constraint has come into working memory and by specific

conditions the constraint is concerned with. At any time, a given number of constraints (those that have passed through working memory) will have contributed their utility to particular alternatives, and a given number (those attached to relevant concepts not in working memory) will have not have contributed. In order to make decisions using incomplete information such as this, the agent has in its working memory a constraint on utility, representing the minimum utility necessary for an alternative to be acted upon. This constraint may be global or local to a particular activity the agent is performing, and serves to limit deliberation by allowing the agent to select the most important alternatives. It is not static, and can be affected by intentions and concepts in working memory as well as higher-level knowledge. For example, when conflicting intentions are present, high-level knowledge may alter the agent's utility constraint to be higher in order that the agent have greater confidence that it is selecting the right action in these situations. If no action can be performed, the agent may wait for more information to be processed through working memory. This affords the possibility of a higher utility for one or more alternatives, or for constraints in working memory to alter the agent's utility constraint, allowing less highly-ranked alternatives to be selected. It also presents the possibility that the agent could miss some time-constrained opportunity. Utility may also be used in conjunction with activity-specific measures, as described in [Anderson, 1995; Anderson and Evans, 1996a]. Detailed examples of the use of this control scheme may also be found in these works.

Constraints are also used to control many other aspects of this architecture, such as memory management. The most appropriate memory management policy varies with the situation (e.g. if the agent is in a time-constrained situation with regard to some task, working memory should be biased toward concepts contributing to this task). Constraints associated with intentions may suggest an appropriate policy, or more likely, higher level constraints (associated with the setting or more general background knowledge) will recognize when specific policies are required. Constraints also define a perceptual focus for the agent, and can also focus the agent toward retrieving particular concepts from long-term memory and thus follow a particular line of reasoning. Further details on the constraint-directed reasoning mechanisms internal to these agents may be found in [Anderson, 1995; Anderson and Evans, 1996a].

At a surface level, this approach is most immediately compared to that of Bratman et al [1988] and more recent derivatives of in its use of intentions and in the ability to derive new options and deliberate about them. There are several significant differences. First, intentions in this approach are much more sophisticated than those used by Bratman et al. A constraint-directed improvising agent’s intentions serve as active guides: they contribute alternatives for action and connections to background knowledge that can indirectly supply alternatives, as opposed to influencing activity through a vague sense of commitment to the intention itself. This architecture also provides specific methods for generating and dealing with options as a core of the architecture itself, rather than leaving these and other components as implementation-dependent details. Finally, relying heavily on constraints as a knowledge representation mechanism allows an agent to reason as deeply or shallowly about a particular situation as time allows. Elements of this approach can also be compared to other work in real-time planning, including work in partial universal planning [Dean et al., 1993], case-based activity [Hammond et al., 1990], and others. The interested reader is directed to [Anderson, 1995] for much more more detailed comparisons and analysis.

## 5. Discussion

While it is possible (though certainly not easy) to interface a running program with a simulation tool and thereby support agent decision making in an individual-based model, Gensim is one of very few simulators that attempts to provide sophisticated agent definition facilities within the tool itself. More important than making the development process user-friendly however, the previous section has shown the constraint-directed improvisation approach to agent design incorporated within Gensim to be an integral part of supporting a breadth of agents using a single simulation tool. While it is possible to develop very elaborate reactive/deliberative agents for inclusion in individual-based models using this approach (real-time agents have been developed for simulated cooking and other everyday human domains [Anderson, 1995; Anderson and Evans, 1996a,b]), it is equally viable to use the same approach to quickly produce agents with scaled-down intelligences for simpler environments or to represent individuals of lesser intellect[7]. Moreover, the same approach can be used to define a broad range of agents within the same simulation.

The constraint-directed improvisation approach in tandem with the basic organization of Gensim itself also bring to bear several important methods for working effectively under the computational restraints associated with complex individual-based models. Because of the distributed nature of the intentions that form a basis for improvising agents, and the facilities in Gensim itself for sharing knowledge between agents, it is possible for agents to have shared common concepts in intentions, analogous to the vast shared cultural knowledge that humans possess. This not only allows agents to share some common social aspects, but from the point of view of implementation also allows us to define shared knowledge once only, thus greatly reducing the overhead associated with large numbers of agents. Because of the object-oriented nature of agents in Gensim, a hierarchy of agents can be defined with varying amounts of shared knowledge. This in no way compromises the realism of the simulation: within the improvisational model, agents can be constructed to improvise on the same routines in very different manners simply by changing constraints associated with individual agents. That is, even though a large portion of intentions are shared, because intentions have distributed aspects it is still possible to define differences that alter individual behaviour radically. Given some complex, knowledge-intensive behaviour, the behaviour itself can be represented once at the class or superclass level, and a few simple constraint changes can bias one agent to some aspects of the particular behaviour, another agent to others.

The real power to support agent breadth is derived directly from the constraint-directed approach employed in the agent design. Beyond what is defined in intentions, constraints also completely control each agent's internal mechanisms. This means that a few simple constraint variations between one agent and another can provide enormous individual variations in both intellectual ability and behaviour. Two agents sharing routines, for example, can have different sets of perceptual capacities, different abilities or inclinations to explore background knowledge, and so

---

[7] [Anderson and Evans, 1994] details some simple experiments on improvising agents vs. strict planning and reactive agents in an ecosystem model.

on. This translates into broad variations in individuals with less effort on the part of the developer and less demand on computational resources.

Having expounded on the abilities of this system to support a broad range of agents and a broad range of simulation scenarios in general, it is important to emphasize that despite these developments, Gensim still cannot directly support *every* agent or *every* simulation that could be imagined. While Gensim strives to cover the breadth of simulations and agents depicted in Figure 1, it cannot support every simulation while remaining inside the boundaries of realistic expectations on users depicted in that same figure. To be accurate, there is one exception to this point: Gensim could actually be pushed directly to the extreme of programmability, beyond what we have examined in this Chapter. Given that Gensim is written in LISP, a traditionally interpreted language, its source code is not separated from domains being designed for the simulator as compiled code would be, and a user could indeed reprogram any aspect of the simulator to handle anything that isn't handled already. Whether it would be practical to do so is another matter - this would be moving directly to the right hand side of rest of the spectrum of Figure 2, and building a simulation from scratch (albeit beginning with the bulk of the source code that would be required). While possible, as discussed in section 2, this is not desirable from the points of view of control of experimentation and ongoing changes in agent design.

With regard to restrictions on what Gensim can adequately support, there are several assumptions made regarding the nature of agents in Gensim. As described in the previous section, a Gensim agent must still assume the same timing model as the simulator, and for simulations to operate at various levels of granularity (another important provision for computational feasibility in large simulation models) actions must be defined in a time-independent way. However, as has been previously discussed, these features are necessary and are implemented in such a way as to maximize breadth despite their necessity.

Beyond this, there is one more significant assumption regarding the nature of agents in Gensim. That is, agents as inhabitants of a simulated environment must describe their actions (and their perceptual focus) in a manner that the simulator can comprehend in order to manifest change and/or supply the appropriate perceptual focus. For example, if an agent picks up an object, it must somehow inform the simulator which object the action has been performed with; if the agent wants more information about "that thing over there", it must describe the object as best it can, in order for the simulator to differentiate it from other objects the agent could be referring to and thus respond with the correct information.

This requirement limits agents to an understood grammar and vocabulary of communication with the simulator, and thus places a design assumption on the agents themselves. However, this is no more a problem in Gensim than it would be in any other simulation system, because it is a direct consequence of the simulation problem itself. Communication is one aspect of the relationship between an agent and the real world that can never be completely approximated by any simulator. The reason for this is that no communication takes place when an agent interacts with the real world (at least in the sense that communication is normally thought of). When an agent wants to pick up an object, for example, it just does so: it doesn't have to communicate this fact to the world. This is in part because the world does not exist for the benefit of an agent, the way a simulator does: any agent is simply an object in the world, like any other physical object. It is also due to the fact that the real world keeps track of itself: an agent's actions physically alter

objects, rather than indirectly manipulating some virtual representation of those objects. A simulator, on the other hand, keeps a detailed representation of every object in the world, and changes in the world are manifested by alterations in these abstract entities. Because it no longer occurs naturally, change intended by agents must be communicated to the simulator in some way. A stronger tie between the agent and the rest of the world is thus necessary.

However, in most simulators, this tie and the constraints on agent design it represents, is far too strong. Many simulators do not even use separate representations of objects for agent and simulator: the identical physical chunk of knowledge that describes an object to the simulator also describes it to any agent. When compared to the real world, this is like taking each object in a physical environment, labelling it with a unique agreed-upon symbol, and having everyone reference everything in that fashion.

Like its other facilities, Gensim attempts to take an approach to this problem that creates a compromise between breadth and ease-of-use. The system provides both an indexical-functional or deictic [Agre, 1988] means of describing objects, and also an objective one. The system's deictic representation allows an agent to construct a description from its own perspective in a communication language (something might become, in translation, the-animal-in-front-of-me, or the large-object-to-my-right, for example) which is then parsed by the simulator to determine the object desired. This is realistic from the point of view of human representation [Agre, 1988], but once again is more sophisticated than will often be needed. More importantly, if real-time aspects of the domain are of interest, the time required to construct such descriptions may significantly affect the agent. The objective method, where an agent simply specifies its internal designation for the object (some random symbol name), and the simulator then examines the agent's internal representation to match the object to one in the environment, suffices in these cases. From the point of view of accurate simulation, the latter is still suitable: in no case does an agent assume knowledge of the objective world over and above its own perspective, it simply saves time constructing an artificial object description purely for the simulator. Details of the communication language used may be found in [Anderson and Evans, 1995; Anderson, 1995].

Within the realm of current and future development, there are further aspects of the agent-simulator relationship that can be made more realistic. As stated previously agent vision, for example, is both an active and passive process, meaning that some elements are directly affected by the agent itself (what it's looking for) and others are directly affected by the environment (what can be seen). Currently vision is defined within the agent itself, simply to enforce the separation of agent and simulator knowledge, one of the major design goals of the system. However, like agent-simulator communication, this is somewhat artificial. If an environment is affected in such a way that vision is impaired (e.g. fog) it is partly an environmental consequence (the moisture in the atmosphere interfering with light transfer) and an agent consequence (it's vision can't adapt to these effects). We are working on ways of extending the breadth of agents supportable by defining another level between agent and simulator to cover situations such as this more realistically in situations that require it. This will be handled by an agent-simulator interface defined for each agent, where shared aspects such as visual acuity under environmental conditions can be defined. Once again, this can be shared between agents and classes of agents in order to facilitate the definition and support of significant numbers of agents. Also once

again, it is a method to extend the breadth of support the simulator offers, and can be avoided in simple scenarios where it is not required.

Another issue of importance in agent and simulation breadth is support for social-level simulations. There are several aspects of this: that agents can be formed into groups and manipulated in groups by the simulator; that agents can communicate with one another; and that explicit social-level interactions can be defined as well as being emergent from individual actions. The first two of these can already be handled to varying degrees by Gensim. As agents already communicate with the simulator, precisely the same communication methods can afford inter-agent communication as well. In addition, should other forms of communication be desired, communicative actions can be defined as easily as any other form of action within the system.

With regard to the aggregation and disaggregation of agents, and multiple levels of granularity in general, these aspects are also supported to a significant degree within the existing system. Multiple levels of granularity are supported in part by the object-oriented approach to the simulator itself, in that actions can be defined to affect groups of agents, and that groups of agents can be manipulated as a whole. Granularity in other simulation aspects is also supported by the system. As described in section 2, for example, time can also be implemented to the degree of granularity desired. One general desire in future development is to make the system move between levels of granularity as easily in terms of space as it currently can in terms of objects and time. Given that space is currently either defined as a grid-based mechanism or through specific user-defined locales, it is currently up to the user to perform any necessary spatial aggregation and disaggregation. Full support of this along with ease of use will likely necessitate the interfacing of the simulator and its objects with a more traditional GIS system to provide spatial representation and manipulation. Current work on this facet of the system involves converting Gensim's internal spatial representation to a more Gecko-like [Booth, 1996] representation (effectively a real-based grid structure as opposed to a discrete interval), and removing its dependence on a user-defined set of qualitative locations.

In order to facilitate explicit representation of social interaction, we are extending the improvisational paradigm described in Section 3 to include collective shared intentions, with multiple agents as participants. This is currently supportable within this existing model, but requires explicit user definition of specific communication actions required, as well as the definition of internal agent concepts that go along with social planning: expectations, commitments, responsibilities, etc. Note that these also are easily viewed as constraints and can fit into individual intentions within the member agents contributing to social interactions. It is simply up to the modeller to currently define them, putting this aspect uncomfortably beyond the reasonable boundaries depicted in Figure 1. Extending Gensim to better support this involves separating individual and social knowledge in intentions, defining explicit coordination mechanisms (e.g. a wider array of built-in communication actions) and also defining some of the many implit coordination abilities social animals possess (e.g. interpreting one's own perception of other agents in light of shared intentions). By extending Gensim and its internal support for improvising agents to encompass the definition of these facilities, the breadth of supportable simulations within acceptable ease-of-use boundaries will further be extended.

In combination, the current version of the Gensim simulator and the constraint-based improvisation approach to agent design outlined in this Chapter provide a unique tool for developing individual-based simulation models incorporating intelligent agents. We are still, however, using this tool exclusively in our own agent-based research for several reasons. While completely functional, its user interface is still primitive, as we have been concentrating limited resources on developing realistic agent-environment interfaces and on the flexibility of the agents themselves. The system's timesharing implementation is both Macintosh- and Common Lisp-specific, as are other portions of its system code, and so a major goal in the immediate future is to port the system to a more widely-used platform (Java) before going on to develop such facilities.

While attempting to support breadth in intelligent agents is the obvious contribution of this work, it can be argued that the most significant contribution of this work is similar to that described in the other chapters of this book: to point out the commonalities between research in intelligent agency and that of simulation modelling in general and ecosystem modelling in particular, and to reap the benefits of the synergy that results. This Chapter has expounded on the virtues of intelligent agents and the benefits a broad range of agents can bring to simulation models. These benefits are very promising: the inclusion of intelligent agents can offer improved accuracy of ecosystem models, as well as the ability to include in ecosystem simulations intelligent agents designed to give advice directly to resource managers. However, it is even more readily apparent as an AI researcher that studying how intelligent agents can assist in simulation models will also be of great benefit to the field of artificial intelligence as a whole. Gensim was originally constructed because existing simulation tools did not provide the support needed to construct flexible, complex domains for the purpose of testing intelligent agent designs. The detailed domains that will emerge from intelligent ecosystem modelling will be able to serve as such testbeds, providing a wide variety of complex and dynamic domains for intelligent agent research. The ability to support human-level intelligent agents specifically in simulation using such tools has the potential for great impact on many areas outside of ecosystem management (e.g. [Tambe et al., 1995]). Moreover, the possibility of using such systems for training humans to behave optimally in problem-solving situations has also been stressed as an important future goal for the field of Artificial Intelligence as a whole [Grosz and Davis, 1998]. In addition, some of the issues described in this chapter, such as aggregation and disaggregation in models, are also very important issues in particular simulation applications (e.g. [Hillestad and Juncosa, 1993]) that work such as this can also contribute toward.

The rigorous standards of ecological modelling and the control over experimentation that results will also serve AI well, ensuring that the models used to test intelligent agents are realistic. AI has much to learn about the formal methods of experimentation that fields such as ecosystem modelling hold as central, and it is only comparatively recently that any works have emerged advocating rigour and control in experimentation in AI (e.g. [Cohen, 1995]) Overall, the addition of intelligent agents to complex ecosystem models has the potential to greatly advance both fields. Tools such as Gensim and the others described in this book are a significant step in this direction.

## *Bibliography*


Agre, P. E., 1988. *The Dynamic Structure of Everyday Life*. Ph.D. Dissertation, Department of Electrical Engineering and Computer Science. 282 pp.

Agre, P. E., and D. Chapman, 1987. "Pengi: An Implementation of a Theory of Activity", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, pp. 196-201.

Agre, P. E., and I. D. Horswill, 1992. "Cultural Support for Improvisation", *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, pp. 363-368.

Anderson, J., 1995. *Constraint-Directed Improvisation for Everyday Activities*. Ph.D. Dissertation, Department of Computer Science, University of Manitoba. 389 pp.

Anderson, J., and M. Evans, 1996a. "Constraint-Directed Improvisation", Proceedings of the Eleventh Biennial Canadian Society for the Computational Studies of Intelligence Conference (AI-96), Toronto. *Lecture Notes in Artificial Intelligence 1081*. Springer-Verlag, Berlin, pp. 1-13.

Anderson, J., and M. Evans, 1996b. "Real-Time Satisficing Agents for Complex Domains", *Proceedings , Ninth Florida AI Symposium*, Key West, pp. 96-100.

Anderson, J., and M. Evans, 1995. "A Generic Simulation System for Intelligent Agent Designs", *Applied Artificial Intelligence* 9:5, pp. 527-562.

Anderson, J., and M. Evans, 1994. "Intelligent Agent Modelling for Natural Resource Management", *Mathematical and Computer Modelling* 20:8, pp. 109-119.

Anderson, J., and M. Evans, 1993. "Supporting Flexible Autonomy in a Simulation Environment for Intelligent Agent Designs", *Proceedings of the Fourth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Tucson, Arizona, pp. 60-66.

Bates, J., A. B. Loyall, and W. S. Reilly, 1992. "Broad Agents", *SIGART Bulletin* 2:4

Beer, R., 1990. *Intelligence as Adaptive Behaviour*. Academic Press, Boston. 213 pp.

Boden, M., 1973. "The Structure of Intentions", *Journal for the Study of Social Behaviour* 3:1, pp. 23-46.

Booth, G., 1996. *Swarm Gecko: A 2-D Floating World for Ecological Modelling.* Technical Report, Center for Computational Ecology, Yale Institute for Biosphere Studies. 24 pp.

Bratman, M. E., D. J. Israel, and M. E. Pollack, 1988. "Plans and Resource-Bounded Practical Reasoning", *Computational Intelligence* 4, pp. 349-355.

Carter, J., 1997. *Moab: A Spatially-Explicit, Individual Based, Expert System for Creating Animal Foraging Models*, Poster presentation at Technology 2007, Boston, MA. http://www.abpi.net/T2007/posters/moab.html.

Cohen, P. R., 1995. *Empirical Methods for Aritificial Intelligence*. MIT Press, Cambridge, MA. 560 pp.

DeAngelis, D. L., D. M. Fleming, L. J. Gross, and W. F. Wolff, 1996. "Individual-based Models in Ecology: An Overview." *Proceedings, Third International Conference on Integrating GIS and Environmental Modeling*, http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/main.html.

Deadman, P., and R. H. Gimblett, 1994. "A Role for Goal-Oriented Autonomous Agents in Modeling People-Environment Interactions in Forest Recreation.", *Mathematical and Computer Modelling* 20:8, pp. 121-131.

Dean, T., L. Kaelbling, J. Kirman, and A. Nicholson, 1993. "Planning with Deadlines in Stochastic Domains", *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, pp. 574-579.

Engelson, S. P., and N. Bertani, 1992. *Ars Magna: The Abstract Robot Simulator Manual*, Technical Report, Department of Computer Science, Yale University. 73 pp.

Etzioni, O, 1993. "Intelligence without Robots: A Reply to Brooks", *AI Magazine* 14:4, , pp. 7-13.

Evans, M., J. Anderson, and G. Crysdale, 1992. "Achieving Flexible Autonomy in Multi-Agent Systems using Constraints", *Applied Artificial Intelligence* 6:1, pp. 103-126.

Folse, L., H. Mueller, and A. Whittaker, 1990. "Object-Oriented Simulation and Geographic Information Systems", *AI Applications in Natural Resource Management* **4**(2) 41-47.

Fox, M. S., 1983. *Constraint-Directed Search: A Case Study of Job Shop Scheduling*. Ph.D. Dissertation, School of Computer Science, Carnegie-Mellon University. 184 pp.

Friedland, E., 1977. "Values and Environmental Modelling", in Hall, C. and J. Day, Eds., *Ecosystem Modelling in Theory and Practice.* John Wiley and Sons, New York, pp. 115-132,

Gigon, A., 1987. "A Hierarchic Approach in Causal Ecosystem Analysis", in Schulze, E., and H. Zwölfer, Eds., *Potentials and Limitations of Ecosystem Analysis.* Springer-Verlag, Berlin, pp. 228-244.

Gimblett, R. H., R. M. Itami, and D. Durnota. 1996. "Some Practical Issues in Designing and Calibrating Artificial Human-Recreator Agents in GIS-based Simulated Worlds", *Complexity International* 3.

Grosz, B., and R. Davis, 1998. *A Report to ARPA on Twenty-First Century Intelligent Systems*. AAAI, http://www.aaai.org/Policy/Papers/arpa-report.html.

Hammond, K., and T. Converse, 1991. "Stabilizing Environments to Facilitate Planning and Activity", *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, p. 787-793.

Hammond, K., T. Converse, and C.Martin, 1990. "Integrating Planning and Acting in a Case-Based Framework", *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, pp. 292-297.

Hayes-Roth, B., E. Sincoff, L. Brownston, R. Huard, and B. Lent, 1994. *Directed Improvisation.* Stanford Knowledge Systems Laboratory Report KSL-94-61

Hayes-Roth, F., D. Waterman, and D. Lenat, 1983. *Building Expert Systems*, Addison Wesley, Reading, MA. 444 pp.

Hiebler, D., 1994. "The Swarm Simulation System and Individual-Based Modelling", *Proceedings, Decision Support 2001*, Toronto, pp. 474-494.

Hillestad, R., and M. Juncosa, 1993. "Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models", *Rand Technical Report MR-189-ARPA*. 33 pp.

Jencks, C., and N. Silver, 1972. *Adhocism: The Case for Improvisation.* Doubleday, New York. 216 pp.

Judson, O. P., 1994. "The Rise of the Individual-Based Model in Ecology." *Trends in Ecology and Evolution* 9, pp. 9-14.

Kester, K., 1996. *Individual-based Simulation Modelling: Approaches and Application in Insect Behaviour and Ecology*. Symposia at 1996 Annual Meeting of the Entomological Society of America, Louisville. HTTP://www.inhs.uiuc.edu/cbd/ESA-Annual/ESA_WWW_design.html.

McArthur, D., 1980. Decision Scientists, Decision Makers, and the Gap, *Interfaces* **10**(1).

Neisser, U., 1976. *Cognition and Reality*. W. H. Freeman & Co., San Francisco. 230 pp.

Newell, A., and H. Simon, 1972. *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ. 920 pp.

Norman, D. A., 1988. *The Psychology of Everyday Things.* Basic Books, New York. 257 pp.

O'Neill, R., 1975. Management of Large-Scale Environmental Modelling Projects, In Russell, C., Ed., *Ecological Modeling in a Resource Management Framework.* Johns Hopkins University Press, Washington. pp. 251-282

Pollack, M., and M. Ringuette, 1990. "Introducing the Tileworld: Experimentally evaluating Agent Architectures", *Proceedings, Eighth National Conference on Artificial Intelligence*, Boston, MA, pp. 183-189.

Rich, E., and K. Knight, 1991. *Artificial Intelligence*, Second Edition, McGraw-Hill, NY.

Russell, S., and P. Norvig, 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ. 932 pp.

Schmitz, O. J., and G. Booth, 1997. "Modelling Food Web Complexity: The Consequence of Individual-Based Spatially Explicit Behavioural Ecology on Trophic Interactions", *Evolutionary Ecology* 11(4): 379-398.

Scahill, M., 1996. *Individual-Based Simulation of Mountain Gorillas*. http://larch.ukc.ac.uk:2001/gorillas/simulation/.

Simon, H. A., 1982. "Rational Choice and the Structure of the Environment", in Simon, H. A., Ed., *Models of Bounded Rationality*, Volume 2. MIT Press, Cambridge, MA, pp 259-268.

Slothower, R. L., P. A. Schwartz, and K. M. Johnson, 1996. "Some Guidelines for Implementing Spatially Explicit, Individual-Based Ecological models within Location-Based Raster GIS." *Proceedings, Third International Conference on Integrating GIS and Environmental Modeling*, http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/main.html.

Spofford, W. Jr., 1975. "Ecological Modeling in a Resource Management Framework: an Introduction", in Russell, C., Ed., *Ecological Modeling in a Resource Management Framework*. Johns Hopkins University Press, Washington, pp. 13-48.

Tambe, M., W.L. Johnson, R.M. Jones, F. Koss, J.E. Laird, P.S. Rosenbloom, and K Schwamb, 1995. "Intelligent Agents for Interactive Simulation Environments", *AI Magazine* (16)1, pp. 15-40.

Walker, H., and W. Cuff, 1988. Scientists, Models, and Resource Managers, *Memoirs of the Entomological Society of Canada* **143** 11-17.

Waterman, D., 1986. *A Guide to Expert Systems*. Addison-Wesley, Reading, MA. 419 pp.

Wilson, W. G., A. M. de Roos, and E. McCauley, 1993. "Spatial Instabilities within the Diffusive Lotka-Volterra System: Individual-Based Simulation Results", *Theoretical Population Biology* 43, pp. 91-127.