Making Modularity Work: Combining Memory Systems and Intelligent Processes in a Dialog Agent

Joanna Bryson
MIT AI Lab; Cambridge MA 02139; USA
joanna@ai.mit.edu

Abstract

One of the greatest obstacles to designing a mind is the complexity of integrating different process types, time frames and representational structures. This paper describes a methodology for addressing this obstacle, Behavior Oriented Designed (BOD), and explains it in the context of creating an agent capable of natural language dialogue.

1 Introduction

Modularity of some degree and description is almost universally accepted in modern understandings of the operation of human minds. On a physical level, we know a great deal about the different structures and functions of various elements of the central nervous system. The spinal cord, the neocortex, the hippocampus, the amygdala, the cerebellum, the lateral geniculate nucleus, the various sense organs — while our understanding of these systems is not complete, we have begun to know their individual architectures and their contributions to intelligence as a whole. Similarly, we are developing a set of fairly well described psychological modules we know to be at least partly independent — declarative knowledge, motor skills, episodic memory, drives, emotions, perception, recognition.

Controversies surrounding modularity focus not so much on its existence as on its nature and extent. The question of modularity is not whether it exists, but how it is organized. For example, are modules necessarily fully encapsulated (that is, disconnected from each other) as in Fodor (1983) and Brooks (1991), or can they draw information from each other, as indicated by Karmiloff-Smith (1992) or Ramachandran and Blakeslee (1998). In AI, there is a further question — how can we as creators of intelligence create this order?

This paper describes a methodology for addressing this problem, Behavior Oriented Designed (BOD), and explains it in the context of creating an agent capable of natural language dialogue. The goal is to be able to create a system capable of perception and action; of maintaining both local behavioral coherence and global dedication to multiple, possibly conflicting goals; of learning; and, in this case, of expressing itself. This requires the combination of parallelism with ordered sequential behavior, and modularity with coherence. It requires the integration of both memory and intentionality across several different time frames.

This paper begins with a full description of BOD. We then illustrate the methodology by describing the construction of a dialog agent. We have been pursuing research in this area for two very different, but not necessarily exclusive purposes. The first is to leverage the current state-of-the-art in the design of complex agents to the problem of creating tutorial dialogs. The second is to incorporate semantic and syntactic information gathered through statistical natural language processing into an intentional dialogue agent. Both of these projects are work in progress, but both effectively illustrate important aspects the problem of designing a mind. The paper concludes with a brief discussion of what additional features might be required in BOD for constructing a true "mind," and whether BOD is compatible with these extensions.

2 Behavior Oriented Design (BOD)

2.1 Behaviors and Behavior-Based Design

Behavior Oriented Design is a methodology for constructing complex agents. It is designed to be applicable under any number of languages and most popular agent architectures. As can be gathered from its name, BOD is a derivative of Behavior-Based Artificial Intelligence (BBAI) (Brooks, 1991; Maes, 1991; Matarić, 1997), informed by Object Oriented Design (OOD) (e.g. Coad et al., 1997). Behavior-based AI is an approach that specifies that intelligence should be decomposed along the lines of perception and action. Behaviors are described in terms of sets of actions and the sensory capabilities necessary to inform them. This sensing must inform both *when* the actions should be expressed, and *how*. In other words, there are really two forms of sensing: sensing for detecting context, and sensing for parameters and feedback of motor actions.

The central observation of behavior oriented design is that mere sensing is seldom sufficient for either detecting context or controlling action. Rather, both of these abilities require full perception, which in turn requires memory. Perception exploits experience and expectation to perform discriminations more reliably than would otherwise be possible. This observation has two consequences in the BOD methodology. First, memory becomes an essential part of a behavior. In fact, memory requirements serve as the primary cue for *behavior decomposition*, — the process of determining how to divide intelligence into a set of modules. This strategy is analogous to the central tenet of object-oriented design, that process is best described and ordered in terms of state.

Although behaviors should be autonomous in so far as they provide for their own awareness of appropriateness to context, they should not necessarily be so sufficiently informed as to know whether their current operation is in line with the intentions or behavioral context of the entire agent. This is the other consequence of acknowledging the role of expectation in perception. The intentions of the agent are themselves state, and as such are the domain of a separate module dedicated to arbitration between behaviors. This process is known as action selection.

2.2 Action Selection in BOD

Action selection, unless sufficiently informed, turns into a combinatorially explosive search process, such as productive planning (Chapman, 1987). Behavior oriented design addresses this problem in two ways. First, in common with standard BBAI, the behaviors themselves control many of the details of action, thus significantly reducing the potential search space. Second, BOD relies on reactive planning. Reactive planning provides the expertise of experience and expectation in the form of preprogrammed plan elements, which are executed as seems appropriate based on the agent's perceptions. These perceptions are again based in the behaviors. As this indicates, BOD does not take the strictly encapsulated view of modularity, but rather allows for a well-defined interface between behavioral modules. The form of this interface is also taken from OOD. The interface is built of methods on the objects that represent the agent's behaviors.

Reactive planning provides for the sequencing of behavior through the appropriate execution of reactive plans. Reactive plans are not themselves limited to sequential structure, but generally also exploit hierarchy, with a number of different possible plan elements ready for execution at any particular time. Behavior oriented design provides for both of these means of ordering behavior. *Action patterns* are simple sequences of action and sensing primitives. *Competences* are prioritized collections of plan elements, the operation of which will tend to achieve a particular goal. Most plan elements will also carry perceptually-dependent preconditions for determining whether the element can and should operate. When a competence is active, its highest priority element that can operate is activated.

Consider an example in blocks world. Let's assume that the world consists of stacks of colored blocks, and that we want to enable an agent to meet the goal of holding a blue block. The perceptual operations in this plan are based on the visual routine theory of Ullman (1984), as implemented by Horswill (1995). A possible plan would be:

1	(holding block) (block blue)	*goal*
2	(holding block)	drop-held-object
3	(fixated-on blue)	grasp-top-of-stack
4	(blue-in-scene)	fixate-blue

In this plan, the highest priority plan element is at the top: recognizing that the goal has been achieved. The competence will terminate if either the goal is achieved or if no elements can execute. Otherwise, the highest priority element is executed. Consider the case where the world happens to consist of a stack with a red block sitting on the blue block. If the agent has not already fixated on the blue block before this competence is activated, then the first operation to be performed would be element **4**. Otherwise, if for example the previously active competence has already fixated on blue, **4** would be skipped. Once a fixation is established, element **3** will trigger. If the grasp is successful, this will be followed by element **2**, otherwise **3** will be repeated. Assuming that the red block is eventually grasped and discarded, the next successful operation of element **3** will result in the blue block being held, at which point element **1** should recognize that the goal has been achieved, and terminate the competence.

Infinite retries can be prevented through a number of means: either through habituation at the element level, timeouts at the competence level, or through a separate attentional mechanism which is triggered by repeated attempts or absence of change. As this last potential mechanism indicates, BOD's action selection mechanism also provides for the switching of attention between multiple possible drives. Indeed, a parallel mechanism is necessary in order for an agent to be sufficiently reactive to operate in a dynamic world (c.f. Georgeff and Lansky, 1987; Matarić, 1997; Bryson, in press). In BOD's action selection, this mechanism takes the form of a special root competence that keeps track of the basic drives for the agent and monitors them for changes in focus of attention. I refer to systems with these characteristics as having Parallel-rooted, Ordered Slip-stack Hierarchical (POSH) action selection. These attributes have been used in a stand-alone architecture, Edmund (Bryson and McGonigle, 1998; Bryson, 1999b) and incorporated into other architectures: Ymir, a multi-modal character architecture (Thórisson, 1999; Bryson, 1999a; Thórisson and Bryson, in preperation) and PRS, a leading reactive architecture for agents and robots (Georgeff and Lansky, 1987; Bryson, in preperation).

2.3 The Design Process

The analogy between BOD and OOD is not limited to the obvious implied metaphor of the behavior and the object, as discussed in Section 2.1, nor to the use of methods on the behavior objects for specifying the interface to the reactive plans, as explained in Section 2.2. The most critical aspect of BOD is its emphasis on the design process itself. As in OOD, BOD emphasizes cyclic design with rapid prototyping. The process of developing an agent alternates between developing libraries of behaviors, and developing reactive plans to control the expression of those behaviors. As in OOD, BOD provides guidelines not only for making the initial behavior decomposition, but also for recognizing when a decomposition has turned out to be inadequate, and heuristic rules for correcting these problems.

2.3.1 The Initial Decomposition

The initial decomposition is a set of steps. Executing them correctly is not critical, since the main development strategy includes correcting assumptions from this stage of the process. Nevertheless, good work at this stage greatly facilitates the rest of the process.

The steps of initial decomposition are the following:

1. Specify at a high level what the agent is intended to do.
2. Describe likely activities in terms of sequences of actions. These sequences are the the basis of the initial reactive plans.
3. Identify an initial list of sensory and action primitives from the previous list of actions.
4. Identify the state necessary to enable the described primitives and drives. Cluster related state elements and their primitives into specifications for behaviors. This is the basis of the behavior library.
5. Identify and prioritize goals or drives that the agent may need to attend to. This describes the initial roots for the POSH action selection hierarchy.
6. Select a first behavior to implement.

The lists compiled during this process should be made either in a notebook, or better in computer files, which can serve as documentation of the agent. If the documentation is kept in files, the use of a revision control system is strongly recommended, in order to record the project's history.

Experience has shown that documentation is most likely to be maintained if it is in fact a functional part of the code. For this reason, code files for the separate elements of the system should be kept segregated by function. In particular, most reactive architectures require special definitions of the coded primitives. These definitions should be kept in a single dedicated file, and the primitives sorted by the behavior module in which they are implemented. This file being functional is therefore always up-to-date, and a clear and convenient documentation of the interface.

In selecting the first behavior, it is often a good idea to choose a simple, low-level priority that can be continuously active, so that the agent doesn't "die" immediately. For example, on a mobile robot with a speech synthesizer, the bottom-most priority of the main drive hierarchy might be a "sleep" function, which keeps track of the time and snores every 30 seconds or so. This way, the developer has a clear indication that the robot's control has not crashed, but that none of its interesting behaviors can currently trigger.

2.3.2 The Development Process

The remainder of the development process is not linear. It consists of the following elements, applied repeatedly as appropriate:

- coding behaviors,
- coding reactive plans,
- testing and debugging code, and
- revising the specifications made in the initial phase.

Usually only one behavior will be actively developed at a time. Again, using revision control is a good idea, particularly if multiple developers are working on behaviors.

Reactive plans grow in complexity over the development time of an agent. Also, multiple reactive plans might be developed for a single platform, each creating agents with different overall behavior characteristics, such as goals or personality. It is best to keep all working plans in a special library, each commented with the date of its development, a description of its behavior, and a record of any other plan or plans from which it was derived. A library of historic plans can be used as a testing suite if any radical change is made to the behavior library.

Testing should be done as frequently as possible. Developing in languages that do not require recompiling, such as perl and lisp, significantly speeds the development process, though it may slow program execution time.

2.3.3 Revising the Specifications

The most interesting part of the BOD methodology is the set of rules for revising the specifications. As in OOD, one of the main goals of BOD is to reduce redundancy. If a particular plan or behavior can be reused, it should be. If only part of a plan or an action primitive can be used, then a change in decomposition is called for. In the case of the action primitive, the primitive should be decomposed into two or more primitives, and the original action replaced by a plan element. Ideally, the new plan element will have the same name and functionality as the original action. This allows established plans to continue operating without change.

In general, the main design principle of BOD is *when in doubt, favor simplicity.* A primitive is preferred to an action sequence, a sequence to a competence. Heuristics like the above can then indicate when the simple element must be broken into a more complex one.

If a sequence sometimes needs to contain a cycle, or often does not need some of its elements to fire, then it is really a competence, not an action pattern. A competence may be thought of, and designed, as a sequence of behaviors that might need to be executed in a worst-case scenario. The ultimate (last / goal) step is the highest priority element of the competence, the penultimate the second highest and so on. Triggers on each element determine whether that element actually needs to fire at this instant. If a competence is actually deterministic, if it nearly always actually executes a fixed path through its elements, then it should be simplified into a sequence.

Competences are really the basic level of operation for reactive plans, and a great deal of time may be spent programming them. One way a competence can flag a need for redesigning the specification is by relying on large numbers of triggers. Perception should be handled at the behavior level; it should be a skill. A large number of triggers should be converted into a single perceptual primitive. Another problem can be that too many elements are added into the competence. This makes design more difficult by increasing the probability that a design fault might result in plan elements that operate against each other, unsetting each other's preconditions. More than seven elements in a competence, or difficulty in appropriately prioritizing or setting triggers, indicates that a plan needs to be decomposed into two plans. If several of the elements can be seen as working to complete a subgoal, they may be moved into another competence which replaces them as an element of the parent plan. If two or more of the elements always follow each other in sequence, they should be removed and made into an action pattern, which is again substituted into the original competence. If the competence is actually trying to achieve its goal by two different means, then it should be broken into two sibling competences which are both inserted into the competence's parent plan, with appropriate triggers to determine which one should operate.

2.4 Differences from Related Approaches

Given that this section has emphasized the analogies between BOD and other related approaches, it may be useful to also quickly outline some relevant differences. It should be noted first that BOD and its competitors are methodologies, not just algorithms. In most cases, it should be at least *possible* to solve problems under any approach. The difference is how easy (and consequently, how likely) it is to solve problems using a particular strategy.

There are two main benefits of BOD over standard BBAI: BOD's use of hierarchical reactive plans, and BOD's methodology of behavior decomposition.

Having explicit reactive plans built as part of the architecture greatly simplifies control. When one particular set of behaviors is active (say a robot is trying to pick up a teacup) there is no need to worry about the interactions of other unrelated behaviors. The robot will not decide to sit down, or relieve itself, or go see a movie unless it is at a reasonable juncture with the tea cup. On the other hand, it may drop the cup if something truly important happens, for example if it must fend off an attack from a large dog trying to knock it over. It is much easier to express this information in a reactive plan than to build complex mutual inhibition systems for each new behavior every time a behavior, as is necessary in conventional BBAI. In mutual inhibition or reinforcement systems, the control problem scales polynomially, with explicit plans the problem scales linearly.

What BOD offers in terms of behavior decomposition over other BBAI methods is:

- A better place to start. Instead of trying to determine what the units of behavior are, the developer determines what information the agent is going to need. This is one of the chief insights from OOD.

- A better way to fix things. Unlike other BBAI approaches, BOD does not necessarily assume that decomposition is done correctly on the first attempt. It provides for cyclic development and neat interfaces between behaviors and control.

Although BOD is based on OOD, it is not a fully object-oriented approach. OOD tends to be useful for passive reactive systems, but is used less frequently for designing systems that are actively internally motivated. The addition BOD provides over OOD is the reactive plan component. This allows the expression of motivation and priority as part of the organization of behavior. BOD applies techniques for building plans and decomposing behaviors that are analogous to, but not exactly the same as the OOD methodologies for designing object hierarchies. In BOD the behaviors are not hierarchical, the reactive plans are.

Another recently developed methodology is Agent Oriented Design (Iglesias et al., 1999). AOD assumes that every module is an agent, with intentions and fully encapsulated state. This differs from BOD, which allows different behaviors to have access to each other's state, and models intentions and planning on a global level, across behaviors, not within them. AOD is actually more analogous to standard BBAI than to BOD.

3 Behavior Decomposition and Plan Construction for Dialogue

Behavior oriented design was developed as a methodology to address the scaling issue in behavior based artificial intelligence. So far it has been applied to problems in blocks world, mobile robotics (Bryson and McGonigle, 1998), artificial life (an animal in a simulated ecosystem) (Bryson, 1999b), and virtual reality character development (Bryson, 1999a). However, none of these applications illustrate BOD at a level of cognitive complexity that suits the conventional implications of "mind". For this reason, this paper focuses its illustrations on work currently in progress in the application domain of natural language dialogue.

Dialog systems currently require an enormous amount of engineering, and typically result in relatively brittle systems. We are currently exploring the use of reactive planning in general and BOD in particular for simplifying dialogue system design.

3.1 Selecting Initial Primitives: the First Plan

We begin by considering as an example the problem of dialog management in a system such as TRAINS-93 (Allen et al., 1995). This system was a major effort in addressing the complete problem of dialog, including having a system capable of planning and acting as well as discussing its plans and acquiring its goals verbally. The TRAINS system served as an assistant to a manager attempting to make deliveries of commodities, such as bananas and orange juice, to a number of different cities. In addition, various cities had various important resources, such as trains, cars, processing plants and raw commodities. These cities were connected by rail, so transport requires scheduling in both time and space.

To build a dialog system similar to TRAINS-93, we first list a rough set of capabilities we expect the agent will have. In this case, we can use the existing system as a guide, and assume that the agent will eventually need the same set of speech acts as capabilities. While we are organizing the gross behavior of the agent, these speech acts will be simple primitives that merely indicate their place in execution by typing their name. This practice of implementing bare, representative functionality as a part of early design is called *stubbing* in OOD. Based roughly on TRAINS speech acts, the initial list of primitives is the following:

accept or **reject** a proposal by the dialog partner,
suggest a proposal (e.g. a particular engine or location for a particular task),
request information (e.g. a particular of the current plan),
supply-info in response to a request, and
check for agreement on a particular, often necessary due to misunderstandings.

```
(if my-turn)
    (if request-obligation) (if check-request false) reject
    (if request-obligation) (if check-request true) accept
    (if inform-obligation) supply-info
    (if comprehension-failure) check last-utterance
    (if bound-non-requirement)
        (if requirement-checked) check-task
        check-requirement
    (if requirement-not-bound)
        pick-unbound-req , suggest-req
    (if (no task)) request-task
wait
```

Table 1: In this table, indentation indicates depth in the plan hierarchy. Notice that the action primitives generally assume deictic reference, where the perception primitive has set attention to a particular task or requirement.

Working from these primitives, we can construct a high-level plan for dialog management in just a few lines (see Table 1). Here, sensory checks for context are indicated by parenthesis. The primitive actions listed above are in bold face.

The highest level concern for this plan is simply whether the agent should take a turn, or whether it should wait quietly. Once it has decided to take a turn, the highest priority behavior is to fulfill any discourse obligations, including the obligation to try to understand the previous statement if it was not successfully parsed. If there are no existing obligations, the next highest priority is to resolve any inconsistencies in the agent's current understanding, indicated here by having a requirement not entailed by the task bound to some value. This indicates a need either for clarification of the requirement, or of the current task.

If there are no such inconsistencies, but there is an outstanding task to perform, then the next highest priority is to complete the task, which in the case of TRAINS usually involves assigning a particular resource to a particular slot in the problem space. Finally, if there is no task, then this agent, having no other social or personal goals, will seek to establish a new one.

This simple plan indicates a number of elements of state the agent is required to keep track of. These elements in turn indicate behaviors the agent needs to have established. To begin with, the agent needs to know whether it currently believes it has the turn for speaking. Although that may be a simple of bit of information, it is dependent on a number of perceptual issues, such as whether the dialogue partner is actively speaking, and whether the agent itself has recently completed an utterance, in which case it might expect the other agent to take some time in processing its information. The agent may also be capable of being instructed to wait quietly. Further, that waiting might also be time bounded.

3.2 Building a Behavior Library and Drive Structure

To a first approximation, the primitives used in the plan above can be arranged into behaviors as shown in Figure 1.

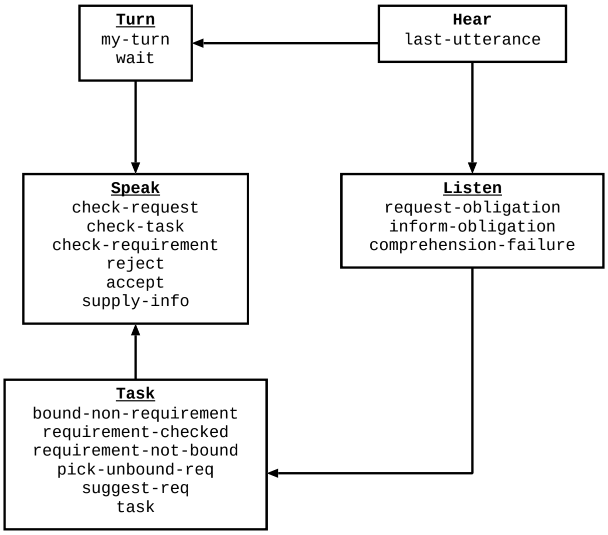

Figure 1: A first cut at a behavior decomposition for a TRAINS-93 type dialog agent. Each box represents a behavior and the primitive senses and actions associated with it. Arrows indicate information flow between behaviors.

The constructive planning required by TRAINS-93 can also be replaced by a fairly short reactive plan (omitted for space) though still supplemented by an $A*$ search algorithm for finding the nearest resources. This suggests that a reasonable initial drive structure for the TRAINS-like dialog agent might be:

1	(if noise)	listen
2	(if need-answer)	think
3	(if my-turn)	take-turn
4		wait

This small plan serves as the parallel operating root of the action selection for the entire dialog agent. The plan in Table 1 fits under the label *take-turn* while the reactive plan for scheduling (including the call to the search algorithm) fits under *think*. A drive structure like this allows another speaker to interrupt, since *listen* has the highest priority. The entire system still relies on the basic behaviors shown in Figure 1. The act of attempting to take a turn would set the flag for "need-answer" if a problem requiring domain-specific planning has been encountered. Solving such a problem should unset the flag, so that turn taking might again operate. Notice that the drive structure has no goal, so will never terminate due to success. Also, the lowest priority element has no precondition, so the drive might never terminate with failure, unless *wait* has a timer and a limit after which *wait* itself fails.

3.3 Scaling the System

The above system obviously hides a great deal of complexity: the problems of parsing the dialog input and constructing sensible output are completely untouched. On the other hand, a BOD system is sufficiently modular that these procedures may be primitives or "black boxes," since many AI systems for language parsing and generation have already been constructed.

Our intention is to use behavior oriented design to organize dialog management for an even more complex system than the one shown above. Our problem domain is tutoring basic electricity and electronics, and we hope to integrate systems that are capable of a wide range of behaviors for assisting students. Examples of desired behavior include analyzing incorrect answers in order to diagnose the learning failure, and providing multi-turn, Socratic method tutoring to lead the students to correcting their basic misconceptions. To be useful with real students, this system will need to be sufficiently reactive to allow the student to either solve the problem prematurely, and also be able to branch into a greater depth of explanation in response to a query or further errors from the student. The design specifications of this tutoring system are described further in (Core et al., 2000).

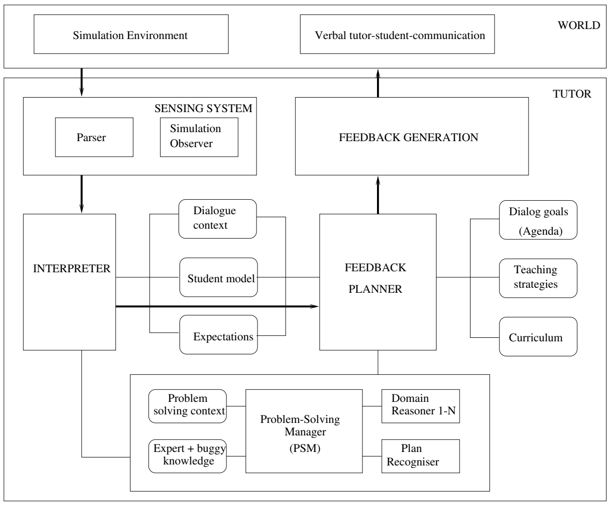

Figure 2: An example of an architecture for a dialogue-based intelligent tutoring system. After Core et al. (2000).

4 Discussion: Can BOD Build a Mind?

The previous sections have presented a methodology for creating complex agents, and an example blueprint for an agent capable of the particularly humanoid capability of natural language dialogue. But can behavior oriented design be used to create a *mind*? This section addresses this question in three different ways. First, we examine the biological plausibility of systems such as those created under BOD. Next, several important elements of mind not explicitly addressed in the previous example are discussed. Finally, we discuss learning.

4.1 Biological Plausibility

BOD hypothesizes the following:

1. most of intelligence is broadly modular,
2. arbitrating between modules requires a specialized mechanism for action selection,
3. complex behavior requires hierarchical and sequential structure for action selection, and
4. switching attention from complex behavior to new salient features or events also requires a specialized mechanism, operating in parallel.

None of these hypotheses have been completely established in the biological literature, however all of them currently have active support in the neuroscience literature. Modularity was discussed in the introduction to this paper. It is also supported by brain cell recording experiments showing that individual cells are associated with different stimuli and/or behavior, and indeed are members of different ensembles, depending on the animal's current context (e.g. Skaggs and McNaughton, 1998). Redgrave et al. (in press) have put forward the hypothesis that the basal ganglia is the specialized organ for action selection in vertebrates. The amygdala has long been implicated as a brain organ dedicated to detecting emotionally salient features in the environment and gating behavior in response to them (Carlson, 1994). Of the hypotheses, the most contentious is probably the third, which I have discussed at length elsewhere (Bryson, 2000). There is considerable evidence that at least some species-typical behavior sequencing is stored in various areas of the vertebrate midbrain (Carlson, 1994).

Another interesting biological analog to the mental architecture constructed under BOD is Rensink's recent theory of vision and visual attention (Rensink, 2000). Rensink proposes that the visual scene is essentially covered with *proto-objects* which are monitored in parallel by the vision system, while only one item is fully attended to at any given time. That item is constructed of approximately four "fingers" of attention which bind proto-objects into the attended, fully represented object. Only attended objects can appear in episodic memory, or be associated with time, though proto-objects may communicate location and gist, particularly on the level of priming. Rensink's work is an interesting parallel, particularly in that it focuses on perception, while BOD focuses on action, and the two models are more or less reciprocal.

4.2 Additional Humanoid Subsystems

As indicated towards the end of the dialog example above, any number of modular systems might be incorporated into a BOD system. This paper and BOD in general emphasize the organization of action and memory. However, a fully humanoid mind might require a number of other systems.

4.2.1 Spatial Action and Perception

An embodied agent, whether embodied physically or in virtual reality, encounters significant challenges not present in a text-based world. For example, a robot typically has noisy and unreliable sensors which require memory and sensor fusion to disambiguate perception. Of course, natural language can also be ambiguous and require multiple knowledge sources. As it happens, BOD's behavior library structure and heuristic programming methodology were originally developed around the problems of mobile robot sensing, and have proven successful in that domain.

Coherent action, particularly of effectors with many degrees of freedom, is easier to perform with an extension to BOD. This extension is a separate intelligent scheduler which selects appropriate motions once the main planner has selected the target positions or gestures. This module is at least roughly functionally equivalent to the cerebellum in vertebrates, in that it handles smoothing of behavior without handling its planning. BOD has been successfully combined with another architecture which has this scheduling feature, Ymir (Thórisson, 1999), although unfortunately very little work has been done so far with this hybrid architecture.

4.2.2 Emotions

Although BOD provides explicitly for motivation and drive, the above system has no specific provision for emotion or affect. In vertebrates, emotions serve as specialized mechanisms for focusing attention, including by deactivating large sections of the cortex (Damasio, 1999). They can also provide complex reinforcement signals for learning behavior (Gadanho, 1999). These functional considerations can be addressed from within BOD. However, a specialized system for mimicking more exactly human or animal emotions would clearly be useful for certain kinds of social interactions — both for making the agent more comprehensible to humans, and for allowing the agent to better model (and therefore perceive) human behavior.

4.2.3 Consciousness and Explicit Knowledge

The dialog system above makes no explicit distinction between conscious and unconscious behavior. Norman and Shallice (1986) propose that consciousness is a sort of special attention which, when activated by some form of interrupt or exception-handling system, aids with a particularly difficult task. This sort of system might be modeled in BOD, perhaps with the more generally associated links to highly plastic episodic memory serving as the specialized attentional systems. BOD systems are capable both of focusing attention on important tasks, and of storing and manipulating records of episodes. However, currently no deliberate model of consciousness has been built under BOD. In fact, the action selection system that operates aspects of the dialog system that are usually unconscious in humans is identical to that which operates elements that are usually perceived as being conscious: the only difference is which behaviors' elements are being manipulated.

4.3 Learning

However powerful a design methodology is, it would always be useful to automate at least some part of the design process by using machine learning. BOD's architecture provides bias useful for enabling certain kinds of learning. Of course, bias and constraint are equivalent: at least one class of problem is very difficult to address within the constraints of the BOD architecture. Nevertheless, BOD can serve as a good starting place for designing minds, including designing minds that learn.

4.3.1 Learning Within Behaviors

BOD is deliberately designed to enable learning within a behavior: in fact, the rate at which state varies is one of the chief cues for what state should be clustered into a particular behavior. Similarly, machine learning techniques can be used for constructing a behavior, or at least part of its state. We are currently engaged in attempting to incorporate statistically acquired semantic lexicons (e.g Lowe, 1997) into a dialogue agent. This could quickly broaden the scope of the agent's ability to recognize conversational contexts. An agent with this lexicon could recognize entire classes of semantically similar sentences for any one programmed interaction.

Similarly, we would like to incorporate the statistically acquired mechanisms of natural language generation of Knight (Knight and Hatzivassilogon, 1995; Oberlander and Brew, in press) into our dialog agent. This would allow us to vary the generative output of our system to be appropriate for various audiences simply by training the mechanism on an appropriate corpus.

Ultimately, it would be interesting to attempt to learn dialog patterns directly from corpora as well. In this case, we could create a "learning Eliza" with only basic turn-taking mechanisms built into the system. The system might be vacuous, but this might not be apparent in gossip-level conversations. We believe, for example, that it might be possible to simulate an individual suffering from William's syndrome this way.

4.3.2 Learning New Plans

Facilitating the learning of new reactive plans was another of the design intentions of BOD. However, this intended feature has not yet been exploited in practice. Learning action patterns should be possible, provided their representation is inserted into a behavior rather than left in a privileged location of the architecture. Learning could occur by trial and error, with generation of new trials done by mutation or recombination of existing patterns. New patterns could also be provided socially, either by imitation or instruction, or by informed search as in conventional planning.

4.3.3 Learning New Behaviors

Although learning new behaviors is clearly a human capacity, this ability is the one most firmly outside of the BOD specification. Allowing for behavioral change would require a complete re-representation of behaviors into some form of generic object class, or better a generic, fine-grained substrate. However, learning acquired modularity in neural representations is currently only in its infancy. There will probably be a significant interval before such techniques can model the complexity of behavior shown in BOD systems. Even if these techniques are finally developed, BOD or a similar architecture might be used to develop the initial agent with its first set of behaviors and competences.

Acknowledgments

Thanks to Jered Floyd, Lynn Stein, Johanna Moore, Claus Zinn, Peter Wiemar-Hastings, Massimo Poesio, Will Lowe and particularly Mark Core.

References

James F. Allen, Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel Martin, Bradford Miller, Massimo Poesio, and David R. Traum. The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):7–48, 1995.

Rodney A. Brooks. Intelligence without reason. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pages 569–595, August 1991.

Joanna Bryson. Creativity by design: A behaviour-based approach to creating creative play. In Frank Nack, editor, *AISB'99 Symposium on Creativity in Entertainment and Visual Art*, pages 9–16, Sussex, 1999a.

Joanna Bryson. Hierarchy and sequence vs. full parallelism in action selection. In Daniel Ballin, editor, *Intelligent Virtual Agents 2*, September 1999b.

Joanna Bryson. The study of sequential and hierarchical organisation of behaviour via artificial mechanisms of action selection. MPhil Thesis, University of Edinburgh, 2000.

Joanna Bryson. Fundamentals of reactive planning. in preperation.

Joanna Bryson. Cross-paradigm analysis of autonomous agent architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, in press.

Joanna Bryson and Brendan McGonigle. Agent architecture as object oriented design. In Munindar P. Singh, Anand S. Rao, and Michael J. Wooldridge, editors, *The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97)*. Springer-Verlag, 1998.

Niel R. Carlson. *Physiology of Behavior*. Allyn and Bacon, Boston, 5 edition, 1994.

David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.

Peter Coad, David North, and Mark Mayfield. *Object Models: Strategies, Patterns and Applications*. Prentice Hall, 2nd edition, 1997.

M. G. Core, J. D. Moore, C. Zinn, and P. Wiemer-Hastings. Modeling human teaching tactics in a computer tutor. In *Proceedings of the ITS'00 Workshop on Modelling Human Teaching Tactics and Strategies*, Montreal, 2000. under consideration.

Antonio R. Damasio. *The Feeling of What Happens: Body and Emotion in the Making of Consciousness*. Harcourt, 1999.

Jerry A. Fodor. *The Modularity of Mind*. Bradford Books. MIT Press, Cambridge, MA, 1983.

Sandra Clara Gadanho. *Reinforcement Learning in Autonomous Robots: An Empirical Investigation of the Role of Emotions*. PhD thesis, University of Edinburgh, 1999.

M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, 1987.

Ian D. Horswill. Visual routines and visual search. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, August 1995.

C. A. Iglesias, M. Garijo, and J.C. Gonzalez. A survey of agent-oriented methodologies. In J.P. Müller, M.P. Singh, and A.S. Rao, editors, *The Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL98)*, pages 185–198, Paris, 1999. Springer Verlag.

Annette Karmiloff-Smith. *Beyond Modularity: A Developmental Perspective on Cognitive Change*. MIT Press, Cambridge, MA, 1992.

K. Knight and V. Hatzivassilogon. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguists (ACL95)*, pages 252–260, 1995.

Will Lowe. Meaning and the mental lexicon. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, August 1997. Morgan Kaufmann.

Pattie Maes. The agent network architecture (ana). *SIGART Bulletin*, 2(4):115–120, 1991.

Maja J. Matarić. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental & Theoretical Artificial Intelligence*, 9 (2/3):323–336, 1997.

Donald. A. Norman and Tim Shallice. Attention to action: Willed and automatic control of behavior. In R.Davidson, G. Schwartz, and D. Shapiro, editors, *Consciousness and Self Regulation: Advances in Research and Theory*, volume 4, pages 1–18. Plenum, New York, 1986.

Jon Oberlander and Chris Brew. Stochastic text generatin. *Philosophical Transactions of the Royal Society of London, Series A*, 358, in press.

V. S. Ramachandran and S. Blakeslee. *Phantoms in the brain: Human nature and the architecture of the mind*. Fourth Estate, London, 1998.

P. Redgrave, T. J. Prescott, and K. Gurney. The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience*, in press.

Ronald A. Rensink. The dynamic representation of scenes. *Visual Cognition*, 7:17–42, 2000.

W.E. Skaggs and B.L. McNaughton. Spatial firing properties of hippocampal ca1 populations in an environment containing two visually identical regions. *Journal of Neuroscience*, 18(20):8455–8466, 1998.

Kristinn R. Thórisson. A mind model for multimodal communicative creatures & humanoids. *International Journal of Applied Artificial Intelligence*, 1999.

Kristinn R. Thórisson and Joanna Bryson. Flexible behavior-based planning for multimodal characters capable of task-oriented dialogue and action. in preperation.

Shimon Ullman. Visual routines. *Cognition*, 18:97–159, 1984.