11

Reactivity and Deliberation in Decision-Making Systems

Carle Côté

11.1 Introduction

Designing decision-making systems for video games can be quite complex and is typically based on experience, intuition, and continuous refactoring. Over the years, successful games shipped using two main types of decision models: *graphical modeling language-based* decision models such as finite-state machines (FSMs), hierarchical finite-state machines (HFSMs), and behavior trees (BTs), and *symbolic planning language-based* decision models such as goal-oriented action planners (GOAP) and hierarchical task networks (HTNs). Unfortunately, little literature exists that explains how and when each approach should be used and for which family of architectural problems they are best suited.

This chapter will present a collection of considerations and thoughts about **Reactivity** and **Deliberation**, two key decision-making system mechanisms. *Reactivity* is about the ability of an agent to be responsive when stimuli are perceived in its environment, while *deliberation* is about the ability of an agent to make decisions and engage consequent actions. Typically, both are required to some extent in the design of every video game agent. By showing how to integrate these key concepts as core design principles, we'll explain how to avoid common pitfalls and create more scalable and flexible decision-making systems.

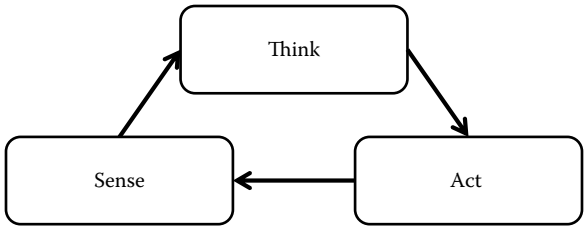

Figure 11.1

Sense-Think-Act paradigm.

11.2 Let's Begin at the Beginning

At its simplest expression, a typical decision-making system revolves around the Sense–Think–Act model, as shown in Figure 11.1. It describes that an agent needs to gather information from its environment (Sense), use the collected information in some decision process to decide what to do next (Think), engage new actions accordingly (Act), and repeat these steps over and over to create autonomy.

While this model shows a very intuitive relationship between an agent's "inner self" and its environment, it doesn't describe anything about the nature of the decision-making mechanisms involved in creating adapted behaviors. In fact, most games require at least two main decision-making mechanisms: reactivity and deliberation. The next sections describe in more detail what they are and their specific roles in a decision-making system.

11.2.1 What Is Reactivity?

Reactivity is the ability of an agent to be responsive to stimuli perceived in its environment. Most of the fun in video games comes from the fact that agents will react to the player's presence—either from direct perceptions (e.g., seeing the player) or indirect perceptions (e.g., hearing broken objects crashing on the ground). In order to be responsive, an agent must engage certain actions in a very short time, otherwise it would create behavioral artifacts that could be perceived by the player as either not believable or not challenging enough compared to their own abilities. These specific actions are called "reactions" and are, as we will see later, crucial when designing decision-making systems. Figure 11.2 illustrates a behavioral timeline with typical examples of reactions.

Reactions can be classified in two categories: *involuntary reactions* and *cognitive reactions*. Involuntary reactions refer to a body's uncontrolled reactions to some events

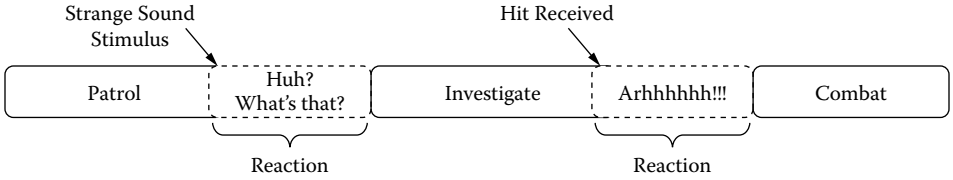

Figure 11.2

Behavioral timeline with reactions.

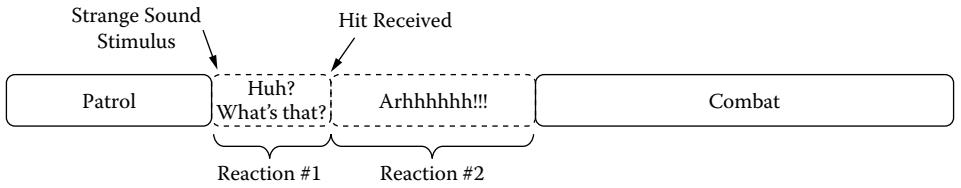

Figure 11.3

Behavioral timeline showing a reaction within a reaction pattern.

like pain, suffocation, or sneezing. Cognitive reactions are reactions that require a minimum of contextual interpretations to be triggered. For example, a loud sound will probably attract attention in a library but would probably be expected on a construction field.

According to our definition, a reaction can have a very short duration (e.g., split second necessary to reorient an agent toward the player) or it can last for a while (e.g., an agent in pain for 10 seconds). This means that during a reaction, other stimuli can be perceived which can potentially create other reactions as illustrated in the timeline in Figure 11.3.

11.2.2 What Is Deliberation?

Deliberation is the ability of an agent to take into consideration many elements of knowledge to decide what to do next. In video games, it's mostly what defines agent behavior in every situation, from high-paced action situations (e.g., combat) to more strategic situations (e.g., investigating an area). In fact, deliberation includes all the rational and irrational introspection mechanisms necessary to execute any tasks. For example, it includes information analysis process, intuition, past experiences, emotions, thinking process, random evaluation, logic, etc. In this article, we'll use the term "decision model" to refer to all these types of introspection mechanisms.

A decision model can take split second, seconds, minutes, days, or even years. For video games, most of the decisions are generally done in seconds or less.

11.3 Common Pitfall #1 : One Decision Model to Rule Them All!

A typical approach to support reactivity in decision-making systems is to use a decision model that can fulfill both reactivity and deliberation requirements. Although this approach seems tempting, it has many drawbacks that are discussed in the following sections.

11.3.1 Different by Nature

Based on reactivity and deliberation definitions, it is effectively possible to conclude that reactivity and deliberation can be merged together as long as the deliberation implementation allows for taking decisions and engaging actions in a very short time. However, this conclusion only considers the responsiveness aspect of the decision model itself; it doesn't consider what triggered the need for a decision and the dynamics of the engaged actions itself. So, let's analyze this a bit.

The first distinction about deliberation and reactivity is the difference between what triggers the need for new actions in both mechanisms. For reactivity, triggering reactions is caused by interruptions that have a higher priority than what is currently ongoing.

For deliberation, the need for a decision comes from two different sources: (1) the current deliberate action (or reaction) is completed and a new action must be engaged, and (2) the current context is changed in a way that it is invalidating or canceling the current action or plan of actions. While some deliberation decisions must be taken rapidly, others can take awhile without causing any problems. This means that deliberation isn't *only* about responsiveness. On the other hand, it is the *main* aspect of reactivity. In video games in general, deliberation tends to be fast because it isn't fun to see inactive agents in a thinking position for a long period of time before taking any action. It is mainly accepted that agents know instantly what to do and how to do it, without hesitation. Still, the very nature of what triggered the need for a decision in both cases is fundamentally different.

The second observation about reactivity and deliberation's nature is related to the nature of their respective undertaken actions. While it is accepted that an agent knows instantly what to do and how to do it, it is also expected that agents will engage in deliberate actions without changing their minds every second for unapparent reasons. Based on this, we can see a second important distinction between reactivity and deliberation: the goal of reactivity mechanisms is to create instantaneous changes in action to reflect body/environmental awareness, while the goal of deliberation mechanisms is to engage the best action possible that can be sustained for the longest time possible according to the context. While this difference seems pretty subtle, it has a big impact on the reactivity and deliberation decision models.

The example illustrated in Figure 11.4 represents a very simple FSM describing the deliberation decision models of an agent in a typical action game.

This FSM can evaluate and execute transitions instantly, that is, it can be used to support reactivity decision models. The states are describing very high-level behaviors like chasing a threat, patrolling the environment, engaging combat, and fleeing a threat. These behaviors can be decomposed in many sub-actions, but they aren't part of the deliberation

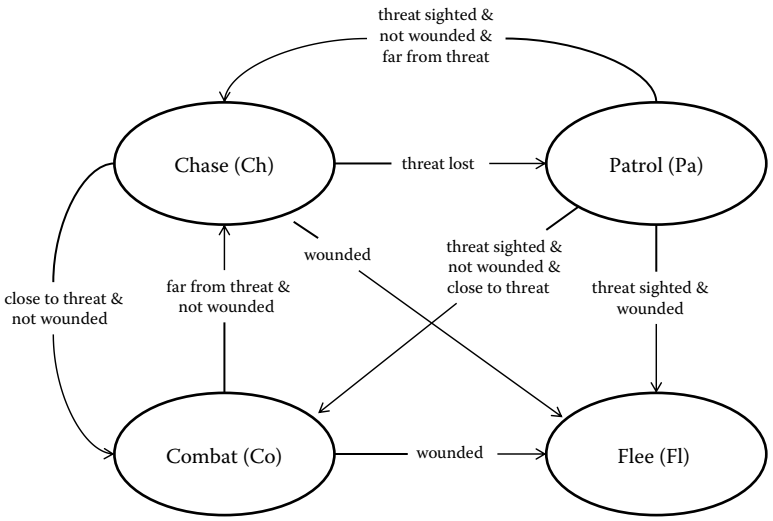

Figure 11.4

Example of a simple FSM describing the deliberation decision model of an agent.

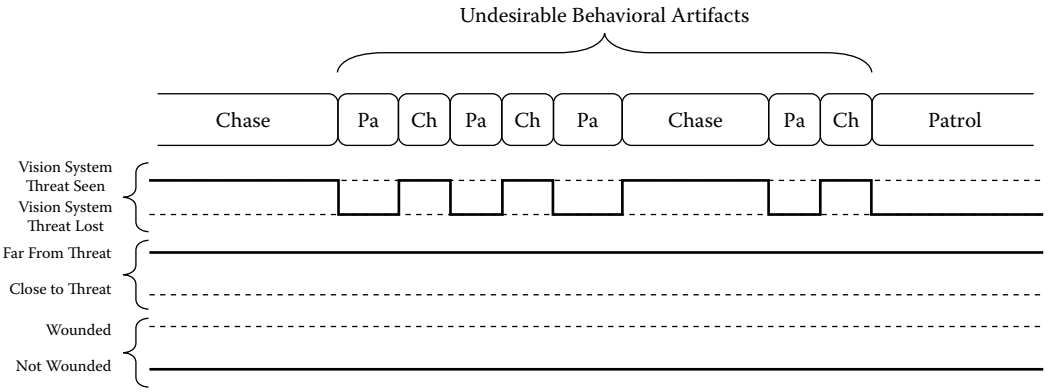

Figure 11.5

Behavioral timeline showing undesirable behavioral artifacts.

decision model expressed into that FSM. The transitions are described using symbols that are also representing high-level concepts such as seeing a threat, being wounded, or being close or far from a threat. It's important to note that no reactivity decision model is shown in this example.

To understand the proper dynamic of this FSM, we need to understand how the agent perceives and analyzes information from its sensors and what it means for this agent to be wounded. For this example, we'll focus on the perception system and presume that the transitions' *threat sight* symbol is directly hooked to the vision system. Figure 11.5 shows a timeline of a situation where the agent can momentarily lose sight of the threat during a chase because of objects preventing the agent to see the threat at all time.

By looking at the timeline, we can observe a lot of transitions in the behavior track. They represent the agent changing its stance from running at the threat to a slow-paced patrol stance multiple times within a couple of seconds because of the vision system losing direct line of sight with the threat. From the player's perspective, the behavior transitions would seem off, and they would most likely be judged as undesirable behavioral artifacts caused for no apparent reason. This is without mentioning that the animation system might not even be responsive enough to execute these fast stance transitions without creating animation popping artifacts. This is a good example to show where responsiveness isn't the only criterion that needs to be considered by deliberation decision models; sustaining actions for the proper amount of time is also crucial to delivering believable behaviors.

In this case, we can solve this issue by hooking the transition's *threat-sighted* symbol to a logical representation of seeing/losing a threat in a chase that would include some form of filtering (using hysteresis algorithms or other similar methods) to avoid creating undesirable oscillations. Figure 11.6 shows an ideal version of the timeline resulting from that logical representation.

11.3.2 Hard to Unify

Considering the different natures of the reactivity and deliberation mechanisms, trying to unify them is very challenging: reactivity is mostly about interruptions while deliberation

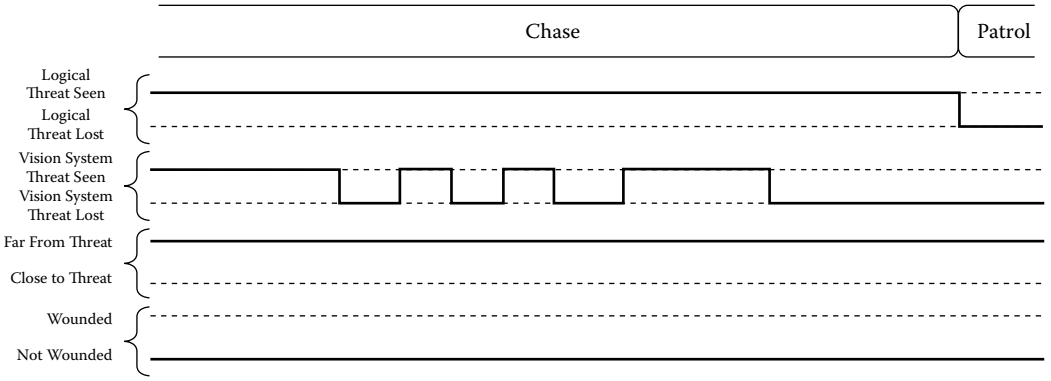

Figure 11.6

Behavior timeline showing the usage of a logical data representation to avoid behavioral artifacts.

is mostly about sustaining states. Any attempt to conciliate them using a unique model is trying to represent conceptual antipodes.

Figure 11.7 represents the same FSM example presented in Section 11.3.1 but including two reaction states: *Hurt* and *Suffocate*.

Because *Chase*, *Patrol*, *Combat*, and *Flee* are deliberation states that are designed to be active as long as possible, they are susceptible to be interrupted at any time. This explains why, in Figure 11.7, we can see that every deliberation state has transitions to every reaction state. Consequently, adding new reaction states to the model would require new transitions from all of the existing deliberation states. The same applies when adding new deliberation states to the model. With increased complexity, it's easy to see that the model will be hard to understand and maintain mostly because it tries to mix two very different kinds of transition dynamics within the same model. To solve this issue, it would be interesting to consider using multiple decision models that can interact together.

11.3.3 Using Multiple Decision Models

It is possible to avoid the limitation of using only one decision model. Figure 11.8 shows an architectural solution allowing multiple decision models. The design principle is pretty simple: create a module (Action Selector) responsible to act as a selector switch between Deliberation and Reactivity modules.

With this architectural solution, Deliberation and Reactivity modules can use their own decision models as long as they can both receive the same stimuli and output their respective set of actions. For example, the Deliberation module could use the FSM presented in Figure 11.4, while the Reactivity module could use a very simple set of rules or a decision tree to evaluate which reaction should be requested according to perceived stimuli. As for the implementation of the Action Selector itself, it can also be done with its own decision model as long as it's able to signal the Deliberation module when a new decision must be taken or to cancel the current deliberate action in order to execute a reaction. Figure 11.9 shows the resulting timeline.

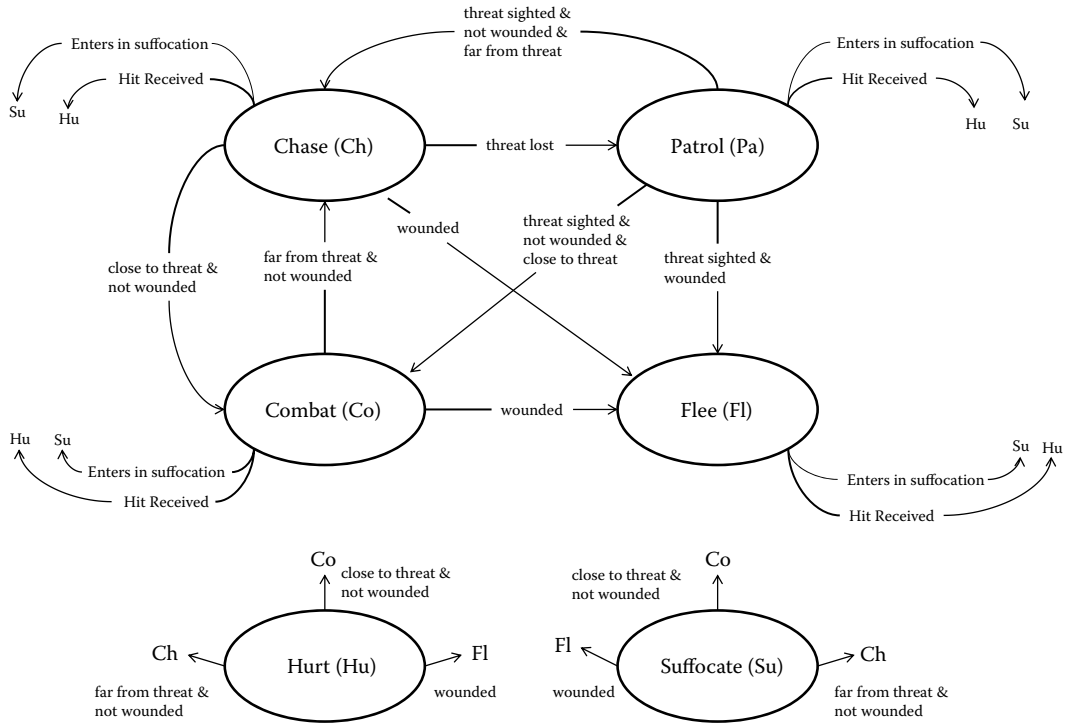

Figure 11.7

Example of an FSM including reaction states.

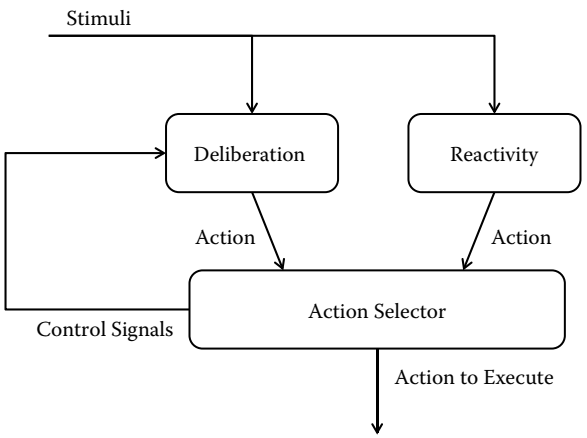

Figure 11.8

Using a selector switch between Deliberation and Reactivity modules.

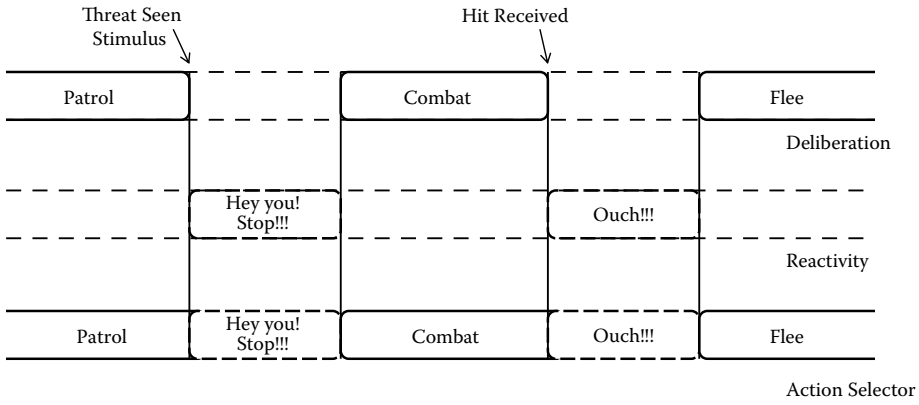

Figure 11.9

Behavioral timeline showing the results of the Action Selector.

11.4 Common Pitfall #2 : One Conceptual Model to Rule Them All!

Another common pitfall is that reactivity and deliberation mechanisms are implemented using the same conceptual model to describe how an agent must behave in every situation. A conceptual model comprises all of the required concepts' definitions and their static/dynamic relationships to create a decision-making system. In fact, when looking closely at each mechanism, we can see many important distinctions that are discussed in the following sections.

11.4.1 Awareness versus Procedural Knowledge

Reactivity is concerned with "danger awareness," whereas deliberation deals with procedural knowledge. Reactions play an important role in a decision-making system that is trying to mimic the physiology associated with the body's inner mechanisms towards self-protection. Two main categories of reaction are presented in Section 11.2.1: involuntary reactions and cognitive reactions. Involuntary reactions are typically created by the body to motivate the individual to withdraw from a dangerous situation. Cognitive reaction is the proactive counterpart where an individual will react preemptively before something can threaten its physical integrity.

By looking closely at these physiological phenomena, we can extract interesting design requirements:

- *Involuntary reactions have a higher priority level than cognitive reactions*. Reactions due to taking physical damages like pain, suffocation, or burning have precedence over any preemptive reactions.
- *Simultaneous involuntary reactions can be combined*. Different physical damages can be received at the same time resulting in simultaneous involuntary reactions. For example, an individual suffocating can simultaneously be hurt by a ranged weapon.
- *Some involuntary reactions have higher priority than others*. Some critical physical damages like being blown away by an explosion or being in heavy pain have precedence over less critical ones.

- *Cognitive reactions depend on the level of danger awareness.* Depending on whether an individual expects danger or not, it might or might not be reacting to some stimuli. For example, hearing a loud broken object sound in the middle of a brawl won't surprise anyone, while it might create a huge surprise reaction in a quiet classroom.

Deliberation isn't tied the same way to the notion of danger awareness. In fact, deliberation is taking this danger awareness notion into account along with many other notions to execute tasks that are important but not necessarily endangering an agent. This means that deliberation's main focus is knowledge and, more precisely, procedural knowledge. Procedural knowledge is the knowledge required to perform any task. When programming an agent to do tasks in its environment, a programmer is actually encoding all of its required procedural knowledge using various decision mechanisms. Depending on what the agent is trying to achieve, different conceptual models can be used. For example, an agent will not use the same decision rules when he's involved in a close-combat situation as when he's involved in a ranged-combat situation. Typically, both situations use different concepts to represent what's important in the environment and the best strategies to use.

11.4.2 Using Multiple Conceptual Models

Using different conceptual models generally allows breaking the complexity in simpler models. This means that, by using the right level of abstraction, it should be easier to write simpler rules and less complex code to maintain. The Action Selector presented in Section 11.3.3 is a good example of this approach. In addition to allowing Reactivity and Deliberation modules to use their own decision models, it also allows them to use their own conceptual models independently. Using the Action Selector as a sequencer between Reactivity and Deliberation modules also simplified the Reactivity module implementation by removing most of the dependencies on Deliberation's conceptual model to select which deliberate action should follow every reaction (as illustrated in Figure 11.7).

The same reasoning applies to the implementation of the Action Selector. It can be implemented with a few simple rules because it uses the right level of abstraction. In this case, the Action Selector only needs to share a minimum set of concepts with the Reactivity and Deliberation modules, that is, knowing if a specific action is a reaction or deliberate action. Figure 11.10 shows the expected timeline from the Action Selector according to its conceptual model.

11.4.3 Separating Decision from Execution

As described in Section 11.4.2, procedural knowledge is a key concept when programming an agent to perform some tasks (or actions). In fact, procedural knowledge generally describes two aspects of what is required to perform a task: the decision model to execute the task (e.g., sequence of actions, various options, how to manage events, etc.) and the decision model to manage the task (e.g., starting conditions, canceling conditions,

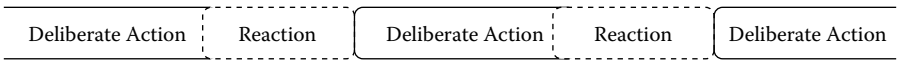

Figure 11.10

Action Selector's timeline.

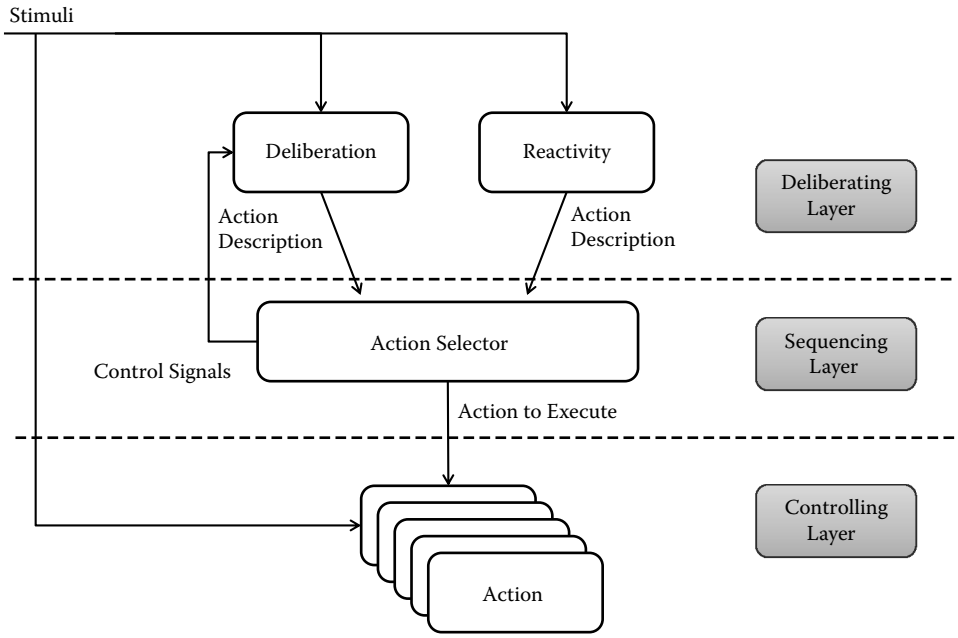

Figure 11.11

An example of a three-layer architecture.

completing conditions, etc). Splitting these decision models can be very useful to reduce the complexity.

In the FSM in Figure 11.4, each state describes a task that an agent should perform if the conditions are met. In fact, it doesn't describe what the agent will do precisely during the execution of this task; it only describes the decision model to manage the task. This means that details of the execution model can somehow be abstracted from the decision model itself without impacting the deliberation mechanisms. This idea has been used many times to create what is called "hybrid architecture" [Murphy 01]. Figure 11.11 illustrates an example of hybrid architecture based on the example presented in Section 11.3.3.

There are only three differences with the example presented in Section 11.3.3. The first one is that the Deliberating Layer explicitly uses decision models to manage actions, instead of the action itself. The second difference is that the role of the Action Selector must not only sequence actions but must also select which actions to execute from the Controlling Layer's action pool. The last difference is the addition of the Controlling Layer, which contains a pool of actions containing the necessary implementations to be executed. Each of these actions can use different decision models and/or conceptual models. They must also be designed to be reactive if required by their respective execution model.

This kind of architecture can be very powerful to use as it offers many ways to break the complexity of a big decision-making system into simpler modules. For example, it can be really easy to change a specific algorithm used to implement an action without impacting other actions or any component from the other layers. Adding a new concept into the

conceptual model of an existing action to enhance its implementation could also represent a very isolated modification to the system.

11.5 Conclusion

This article presented deliberation and reactivity mechanisms as two primary elements to consider when designing decision-making systems. And by understanding their fundamental distinctions, it was discussed that it can be possible to reduce inherent complexity of decision-making systems. Consequently, choosing when to use an FSM, a BT, a Planner, or any other decision models can be a lot easier and based on more solid grounds than pure empirical methods.

References

[Murphy 01] R. R. Murphy, *Introduction to AI Robotics*. MIT Press, Cambridge, MA, 2001, pp. 257–274.