Dedicated to innovation in aerospace

May 2018

Smart Bandits

White Paper

Netherlands Aerospace Centre

NLR

Anthony Fokkerweg 2

1059 CM Amsterdam

p) +31 88 511 3113 f) +31 88 511 3210

1 Introduction

Smart Bandits is NLR's software AI solution for modeling the behavior of simulated forces such as virtual opponents. Starting out as a research project involving multiple PhD students looking into improved methods for behavior modeling, Smart Bandits has grown to become a flexible and reliable tool for modeling behavior. With the help of Smart Bandits, simulated forces can be brought to life in an accessible and transparent manner.

This white paper is intended for readers that are interested in learning more about Smart Bandits. It describes what Smart Bandits is, including the problem it addresses and the motivation for using it (section 1), several example cases for applications (section 2) and technical details about its internal design (section 3). We conclude this paper with future R&D directions (section4). The majority of this paper is non-technical, except for the technical design part in section 3, which is written for a target audience interested from a more engineering-oriented perspective.

1.1 Motivation

In military simulations, computer-generated forces (CGFs) are autonomous entities that represent friendly, neutral or hostile air-, ground- or (sub) surface-based units, platforms or individuals. The behavior that CGFs should display in the simulations is modeled to resemble human behavior, in such a manner that they can express proactive and reactive behaviors autonomously within the dynamic environment in which they are placed. CGFs are typically used in application areas such as training, mission rehearsal, concept development and experimentation (CD&E) or decision-support. For each application, the CGFs require different behavior models. Furthermore, changes in the real world require behavior models to be continuously updated (e.g., in order to rehearse new missions).

Traditional Modeling Techniques

Nowadays the behavioral capabilities of CGFs are quite rudimentary. Their behaviors are often implemented in 'scripts' or based on a set of 'event-actions' rules that are crafted for a specific scenario. They have limited capability to cope with unexpected situations that have not been considered beforehand during the design of a scenario. This makes reuse of behaviors for different scenarios difficult.

The reason for this situation is not necessarily a lack of behavior modeling techniques. Rather, the techniques traditionally offered in most CGF platforms are not suited to create rich, adaptive behaviors. Furthermore, these techniques are often regarded as complicated to work with for end-users (such as operators or instructors) leading to a dependence on specialized personnel for implementing behaviors.

Need for a Paradigm Shift

Because of the continuously growing interest for simulation in the military domain, there has been an increasing demand for more realistic and flexible virtual environments, scenarios and CGFs to simulate missions, tactics, procedures and capabilities of one's own and opposing forces. To keep up with this demand, a paradigm shift is required in CGF behavior modeling going from (a) the more traditional scenario-oriented modeling where CGF behavior is dominantly driven by scenario events and triggers, towards (b) more agent-oriented modeling where CGFs are treated as fully autonomous entities with their own local view of the environment, goals to pursue, and capabilities to plan their actions in a dynamic environment (see Figure 1).

To accommodate this paradigm shift, a new wave of specialized software toolkits has recently appeared within the simulation industry. These toolkits offer more advanced behavior modeling techniques for CGFs to facilitate agent-oriented modeling (e.g. hierarchical finite-state machines, behavior trees and utility-based techniques). Practical versions of these techniques typically originate from the video games industry, which often has been ahead of the simulation industry with respect to quality of graphics, physics, animation and AI.

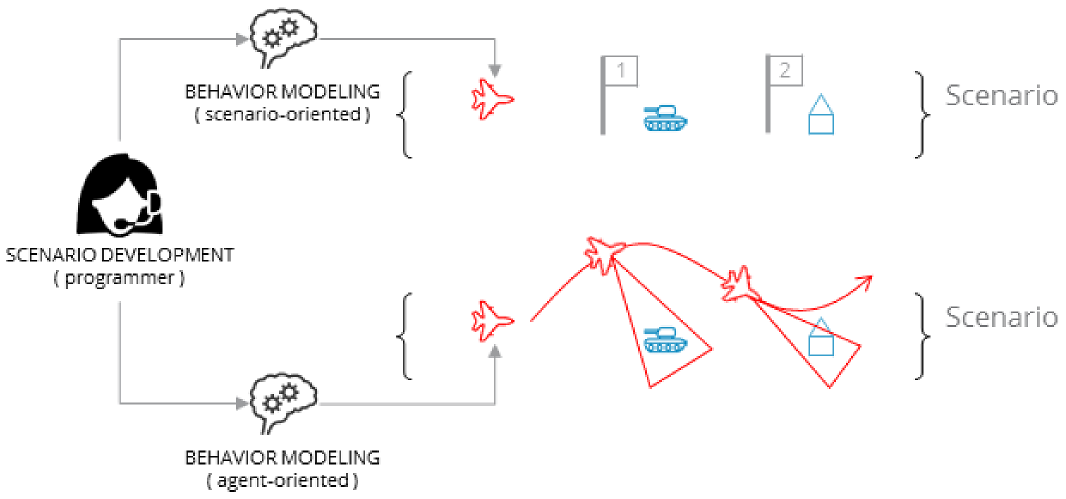


Figure 1. Scenario-oriented and agent-oriented behaviour modeling.

Smart Bandits

Smart Bandits is part of this new wave of agent-oriented behavior modeling toolkits. It can be defined as an AI platform for creating and executing behavior models for CGFs. The key difference with other toolkits is the specific focus on its role as a front-end platform operable by end-users during simulation (e.g. training or experimentation), rather than only as an engineering tool for specialists used during the design phase. This notion is founded in the design philosophy of Smart Bandits which is explained below in more detail.

1.2 Design philosophy

Behavior modeling has always been a rather complicated field of practice. Over the last few decades, a wide range of different modeling techniques have been proposed. Most of them originate from academic research, some of which have become popularized and standardized for use within industry for practical applications (e.g. consider the video game industry). Still, most techniques are reserved only for use by programmers or engineers and few tools exist that make the practice of behavior modeling accessible to people with no or limited technical background. In order to make behavior modeling tools more accessible to different types of professionals, we have given the end-user accessibility of Smart Bandits a central role in our design philosophy.

End-User Accessibility

The key solution for end-user accessibility is to offer the right tools for operators such that they can employ rich behavior modeling technologies without requiring them to take over the role of (and therefore 'become') engineers (see Figure 2). This solution requires a delicate balance: on the one hand, offering a simplistic technique may be very approachable for a wide range of users, but may at the same time hold back expressiveness and the ability to create rich and adaptive models. On the other hand, when offering more advanced and expressive techniques there is the danger that tools become too complicated for users to comprehend and efficiently construct desired behavioral models.

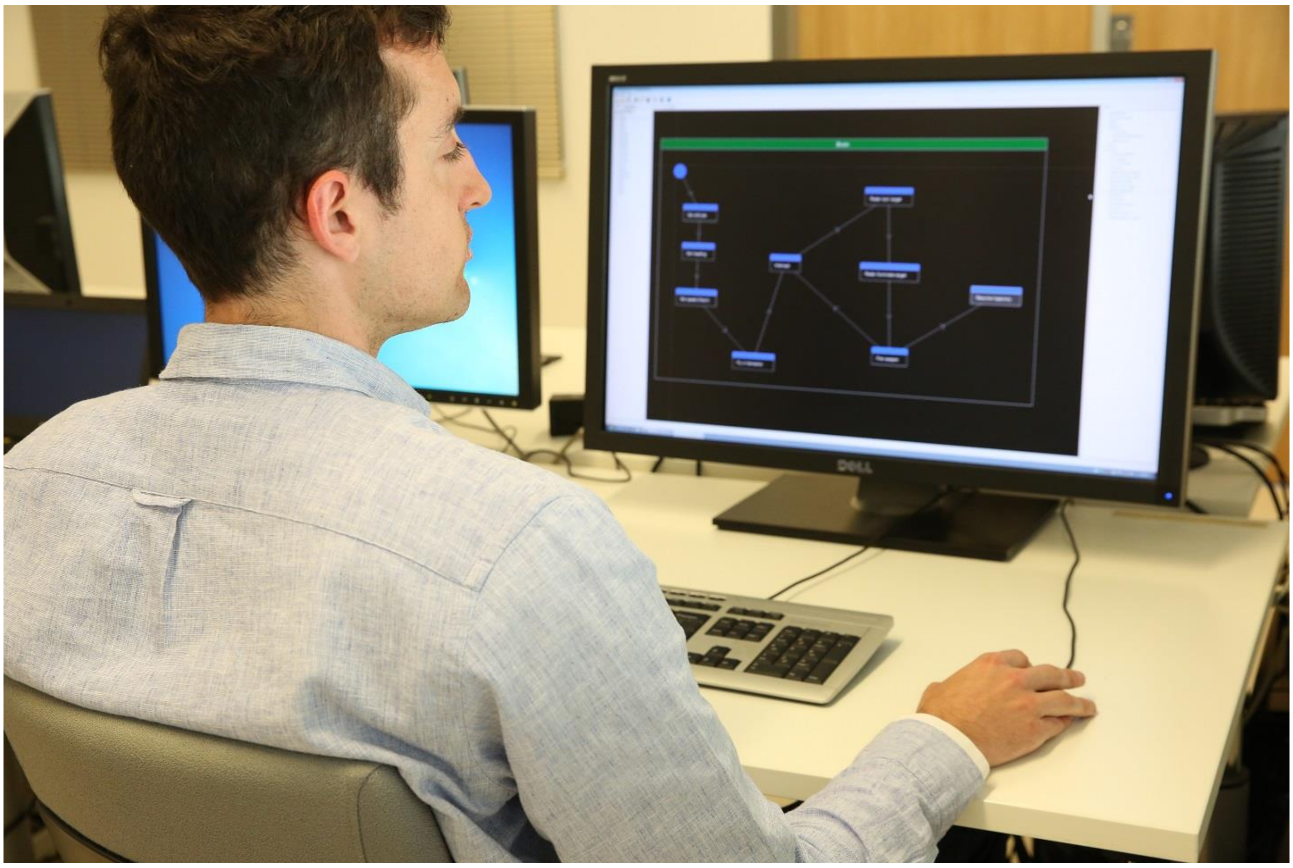
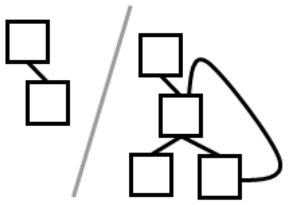


Figure 2: Modeling behavior in Smart Bandits

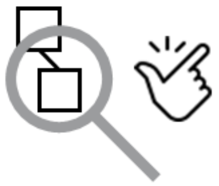
The Design Philosophy of Smart Bandits: Four Foundations

The design philosophy behind Smart Bandits is to find a suitable trade-off addressing the above dilemma. We express this philosophy in the form of four design foundations:

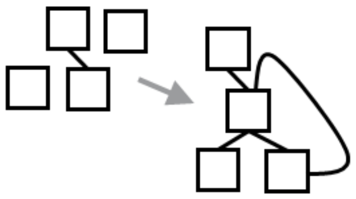
Foundation 1. Smart Bandits can be used as a run-time tool by operators (such as instructors) to define, load, test and run scenarios with agents.



Foundation 2. Smart Bandits can be used to create very simplistic models (e.g. simple behavior scripts) but also rich models that include multi-level tactics and include coordination with other models.



Foundation 3. Operators are able to create and understand behavior models. This implies that no programming knowledge is required and that a user-friendly graphical user interface is offered to make behavior modeling accessible for non-technical personnel.



Foundation 4. Engineers are able to support operators by creating richer, more advanced behavior models in the form of behavioral building blocks that can be reused by operators.

1.3 Key features

Below a summary overview is given of some of the key features of Smart Bandits. Details on the features are described throughout this paper.

Behavior modeling

- Intuitive graphical representation for behavior modeling
- Reusability through behavior templates
- Powerful user-defined knowledge management
- Inter-agent communication and shared situational awareness

Operational use

- Operator-friendly scenario creation and execution
- Real-time behavior prototyping without compilation
- Executable as either a front-end GUI or as a background process (e.g. in batch mode)
- Support for faster than real-time simulations
- Application Programmer Interface (API) provided to enable custom agent extensions
- Suitable for Live-Virtual-Constructive (LVC) and embedded training (ET)

Simulation environment coupling

- Operates independently from the simulation environment
- Works with any environment that exposes its own API
- Network communication interface for distributed simulations
- Readily available integrations for C++, C# and Java-based simulators
- Supports user-defined entity-types, actions and observations

2 Application Cases

Smart Bandits is a general purpose-platform for creating and executing behavior models for autonomous entities (such as Computer Generated Forces) within simulation environments. Therefore, there are no restrictions for its use with respect to type of entities to control (e.g. air, ground or sea platforms) or the purpose of the application (e.g. simulation for training, CD&E or decision-support). Below we present and discuss real-world example application cases in which Smart Bandits provides behavior for CGFs in the areas of training and CD&E.

2.1 Training

One of the primary reasons for using agents (constructive entities) for training is the elimination of human role players. Agents can replace role players to take on the role of team-members or adversaries. The use of agents can also lower the bar for conducting large scale exercises that would not be feasible to arrange with human role players. Finally, training exercises can achieve a more consistent quality of scenarios due to the quality control an instructor has when preparing a scenario with agents and the possibility of reusing validated scenarios.

Figure 3 illustrates an example of using Smart Bandits for training. It combines two training applications, namely a tactical trainer for fighter pilots and a fighter controller trainer.

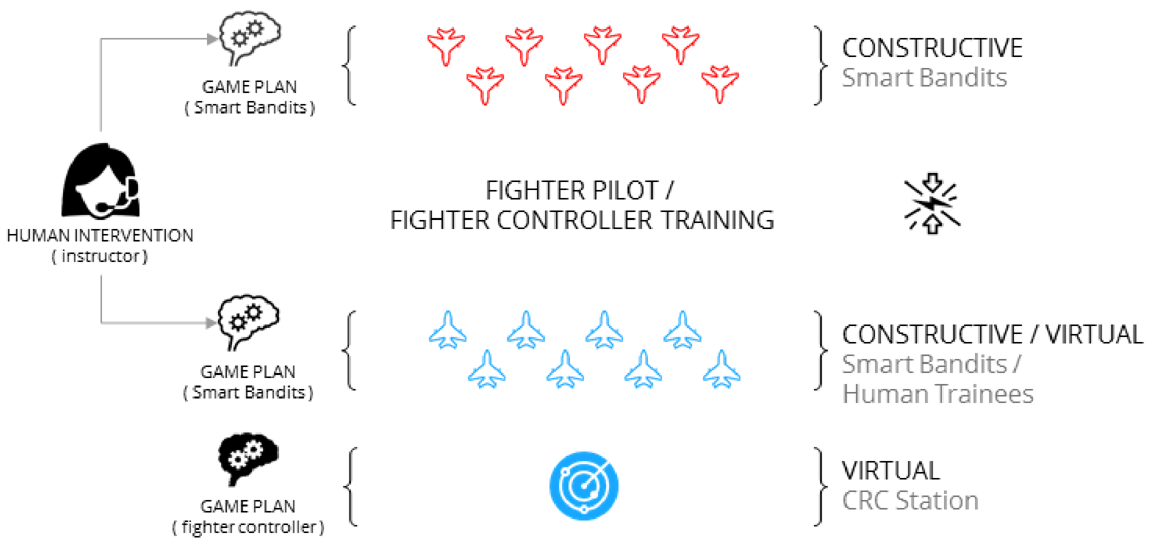


Figure 3: Example Fighter Pilot / Fighter Controller Training

For the fighter pilot trainer, the training environment consists of a single or multi-ship fighter simulator occupied by human pilot trainees, and a set of constructive enemies controlled by Smart Bandits. One can imagine different air-to-air intercept or combat training scenarios with different engagement situations such as 2v2, 2v4 or 4v4. For different scenarios the constructive enemies can be modelled with different enemy game plans or tactics and with different flight formations for approaching the trainee(s).

For the fighter controller trainer, the training environment consists of a terminal operated by a fighter controller trainee, and a training scenario consisting of constructive blue and red forces controlled by Smart Bandits. Similar scenarios configuration can be imagined as described earlier.

In both examples, a human instructor can setup a desired training scenario and oversee the progress of the scenario. If a situation arises where the instructor would like to steer the development of a scenario during training, intervention is possible (within certain bounds) by adjusting the behaviors of the constructive agents in Smart Bandits.

As a real-world example of this use case, Smart Bandits is currently in use by the Royal Netherlands Air Force (RNLAF) F-16 community for operational training. Here, the use of Smart Bandits allows the scenario developers to quickly respond to newly arising training demands.

Of course, while Figure 3 only shows the use of Smart Bandits in a constructive/virtual environment, Smart Bandits is not limited to such a configuration. Smart Bandits is also able to provide behavior for e.g., the CGFs in embedded training systems for live training, given that the appropriate simulation infrastructure is in place. This way, training instructors are able to provide relevant training scenarios to operators in the field.

2.2 Concept Development & Experimentation

Another category of applications for which Smart Bandit agents can be employed focuses on simulations for the purpose of concept development and experimentation (CD&E). CD&E through simulation allows organizations to test and evaluate systems, weapon performance, procedures, tactics and doctrines before they are applied to the real-world. In such simulations agents can be used for two purposes. First they can be used to simulate the system- or human capabilities that are part of the military platform(s) under investigation (e.g. weapon performance or human tactics or rules-of-engagements). Second, they can be used to simulate different course-of-actions of enemy platforms in order to investigate and evaluate a set of so-called *what-if* scenarios.

Figure 4 illustrates an example CD&E application for the naval domain. In this example, experiments are conducted to assess weapon performance and tactics in the domain of anti-air warfare (AAW), anti-surface warfare (ASuW) and anti-submarine warfare (ASW).

As can be seen from the figure, both the own forces and enemy forces are simulated by Smart Bandits. The experiment designer can model and configure the capabilities of the involved platforms to suit the needs of the experiment. Simulations can be run either in real-time or fast-time. Fast-time simulations are especially useful for conducting Monte Carlo simulations where many experiments are conducted with different starting conditions.

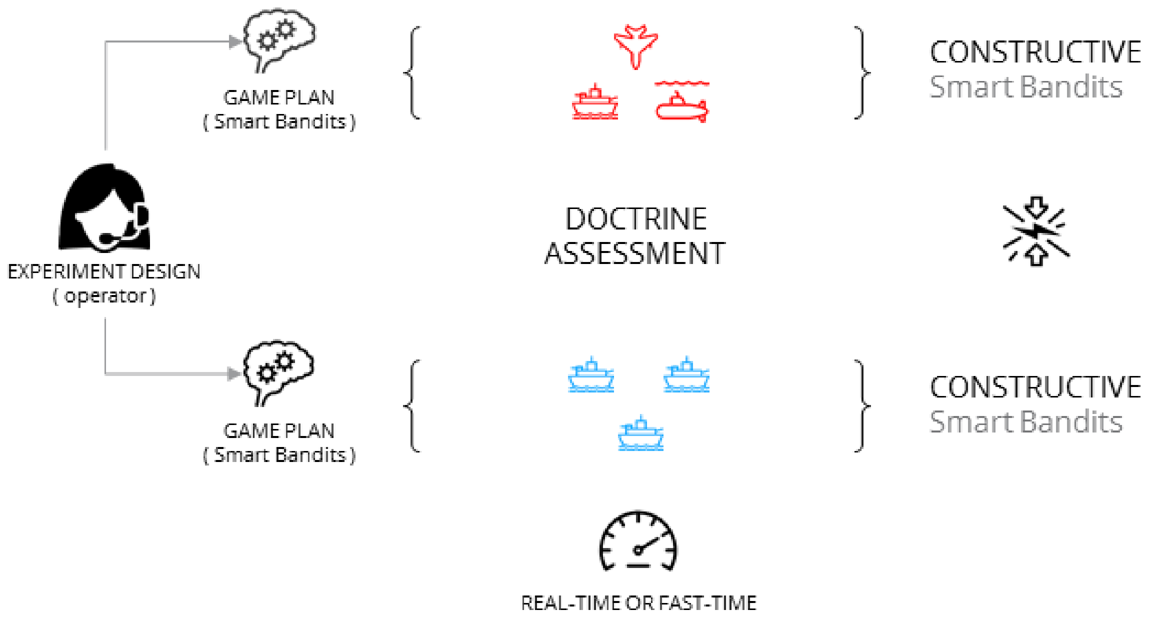


Figure 4: Example Application for Doctrine Assessment

3 Smart Bandits Design

In this section a technical description is given on the internal design constructs of Smart Bandits. It discusses (a) the design from a platform perspective, (b) the interface design with a simulation environment and (c) internal agent design concepts. In the remainder of this section, the terms *agents* and *behavior models* are used interchangeably and denote the same concept. Throughout the discussions examples are given from a single domain for consistency reasons, namely the military domain of air-to-air combat. As a general-purpose behavior modeling platform, Smart Bandits is not limited to this domain.

3.1 Multi Agent System

The Smart Bandits platform at its core can be defined as a multi-agent system (MAS) in software engineering terminology. A MAS is a software system that consists of multiple interacting agents situated in an environment. Agents in a MAS are decentralized, meaning that they have local views of their environment and therefore require communication in order to cooperate or coordinate their activities. A MAS platform (like Smart Bandits) is a software application with a graphical toolkit that facilitates the development of agent models, setting up a connection with a simulation environment, and monitoring of the execution (i.e. behavioral decisions) of agents at run-time.

3.2 Environment interface

A Smart Bandit agent represents a *cognitive* decision-making process. For the model to be executed, it has to be linked to a *physical* entity in a simulation environment. In other words, an agent can be regarded as the *mind* and the physical entity as the *body*. For instance, if an agent is created to control a military entity such as a fighter aircraft or a tank, the agent represents the mind of the pilot or operator. Figure 5 shows the relationship between an agent in Smart Bandits and a physical entity in a simulation engine.

As can be seen in the figure, the interface between an agent in Smart Bandits and an entity in a simulation environment comprises *actions* and *observations*. Actions can be regarded as commands that are sent by an agent to control physical actuators of an entity in the environment. Observations represent information obtained from an entity's sensors and offer the agent situational awareness on which it can base its future decisions. To give an example, actions for a fighter aircraft include controlling navigation (e.g. heading, speed, altitude) and weapon systems (e.g. firing missiles). The observations of the fighter aircraft include visual information, radar contacts, weapon availability, etc. The nature of specific actions and observations depends on the type of entity that is being controlled. For instance, controlling a fighter, a frigate or a soldier would require different type of actions and observations. The choice for a type of entity (and its corresponding actions and observations) depends on the application domain and therefore has to be made by the end-user.

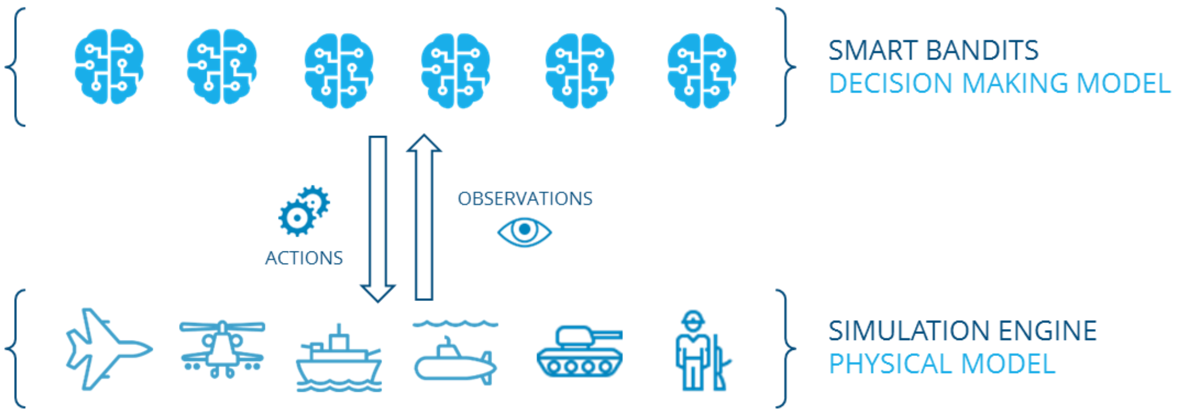


Figure 5: Environment Interface

3.3 Module-based Agent Design

An agent in Smart Bandits consists of *modules.* A module is a processing component that fulfils some functional role of the agent. Two essential functional roles of any agent are *decision-making* (also known as action-selection), and *knowledge management* (or memory management). Smart Bandits is equipped with default modules for these roles. These are explained in the following sections in more detail. Besides the use of default modules, Smart Bandits offers an Application Programming Interface (API) that allows users to add or exchange existing modules with custom modules. To give an example, one could implement a module that performs additional memory processing to simulate a form of memory decay (removing old beliefs), or update old belief as current beliefs based on prediction (e.g. predicting the current position of an enemy aircraft which is no longer in sensory range).

3.4 Decision-making

Decision-making is the process of deciding on a course of action based on an agent's current goals and situational awareness. Many different paradigms exist for decision-making techniques such as scripting, finite state machines (FSMs), behavior trees (BTs), utility-based action-selection or BDI (belief-desire-intention)-deliberation engines. Each technique has its own strengths and weaknesses regarding aspects of expressiveness, modularity and authoring-friendliness.

Currently the decision-making module in Smart Bandits uses the hierarchical FSM (a.k.a. HFSM) paradigm. A HFSM is built up from *states* at which an agent's mental state can reside and *conditions* that define how to transit between different states. One can think of a state as a *situational context* during a mission in which certain goals or plans are being pursued through the execution of actions. Conditions then allow for transition between contexts or a progression of plans based on an agent's situational awareness.

The choice for the HFSM paradigm was made because we consider this paradigm to provide a fair balance between expressiveness, intuitiveness and usability, driven by the design philosophy described in section 0: *behavior modeling in Smart Bandits should be accessible to operators that have no programming or engineering background (e.g. training instructors)*. HFSMs facilitate this through their graphical representations which are easy to read. Additionally, the representation of HFSMs closely resembles how humans tend to explain their line of reasoning when they execute a mission. As an example, Figure 6 shows an illustrative game plan of a fighter pilot (left) and how such a model could be represented in Smart Bandits (right).Some of the drawbacks of FSMs which are commonly mentioned are that of maintainability, scalability and reusability. In Smart Bandits these drawbacks are mitigated through the use of hierarchies, behavior templates and the definition of high-level knowledge. Hierarchies allow users to decompose states into sub-states to introduce different levels of abstractions when modeling behaviors (e.g. decompose a mission into tasks and tasks into sub-tasks). Behavior templates allow users to store pieces of behavior for reuse in other behavior models. Such templates can be parameterized to allow for greater flexibility. Finally, users can define high-level knowledge to represent important situational awareness that can be directly employed in conditions to trigger behaviors or change states. Knowledge management will be described in more detail in the next section.

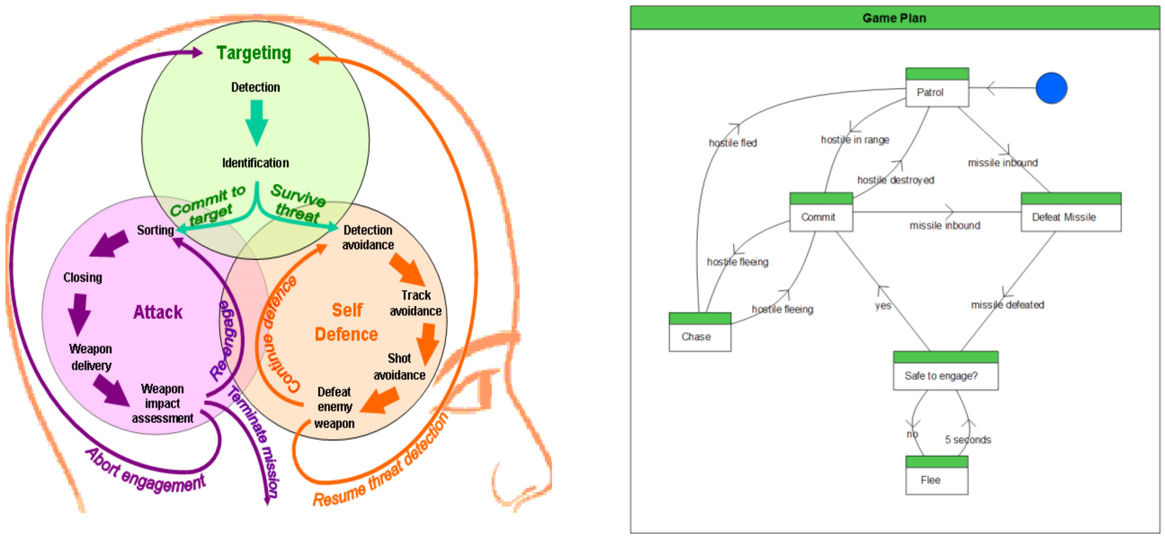


Figure 6: Example fighter pilot model (left) and the model approximated in Smart Bandits (right)

The described decision-making module is the current standard in Smart Bandits. We imagine that future requirements or insights lead to the development of alternative decision-making modules. Thanks to the modular design of an agent, modules can easily be exchanged. For instance, we envision that machine learning modules will play large roles in coming years (see our future directions in section 4). Alternatively, using the Smart Bandits API, users can develop their own modules, either as a replacement of an existing module or as an additional processing module.

3.5 Knowledge management

Knowledge management in Smart Bandits relates to the storage and representation of an agent's knowledge about its environment. This is closely related to situational awareness which concerns the process of perception, interpretation and processing of sensory information from the environment. Having a good situational awareness is crucial for an agent to make intelligent decisions that are in line with its goals, express reactive behaviors and adapt itself to changes in the environment.

Knowledge Storage

Agent knowledge is stored in two forms, namely as most recent *observations* and *entity beliefs*. These two forms can be interpreted as the agent's short-term and long-term memory respectively.

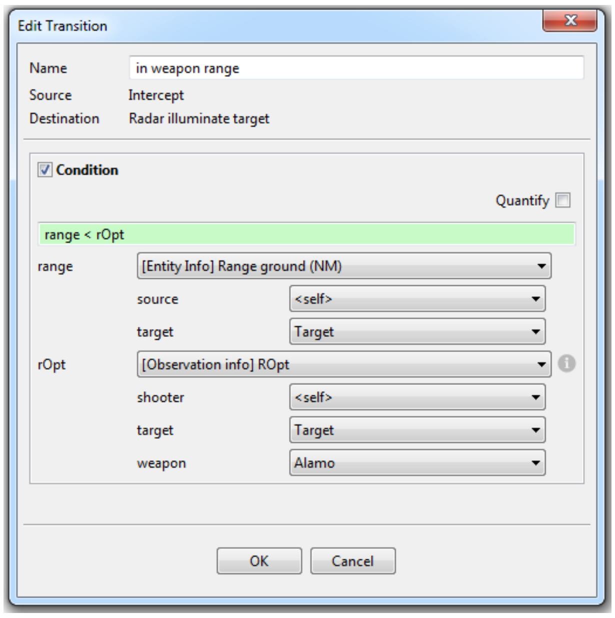
Observations represent the information received from the entity's sensors in the simulation environment. They are typically grouped according to individual sensors and are updated continuously. For instance, common observations for a fighter platform would include information from its navigation systems (e.g. positioning and attitude), information about weapon systems (e.g. number of missiles) and sensor and protection systems (e.g. radar or radar-warning receiver).

Entity beliefs represent persistent beliefs of an agent about itself and other entities in the environment. Entity beliefs are inferred from observations and are timestamped. For instance, information can be stored about positions and headings of enemy fighters, inferred from radar-contacts observations. If the enemies are out of sensory range, beliefs about them can contain their last known positions, so the agent is still able to act on this information.

Knowledge Queries

Knowledge queries are a prime construct in Smart Bandits and are the main method for retrieving knowledge to make decisions. A wide range of knowledge queries are provided. For instance, general-purpose queries can query pre-defined numbers, random numbers, probabilities or the simulation time. Domain-specific queries can reference knowledge about observations (e.g. number of missiles left, whether an enemy is on the radar) or entity beliefs (e.g. the distance between entities, bearings, is an entity an enemy or a friendly).

End-users can use knowledge queries by constructing so-called *expressions*. An expression can be evaluated to a numerical value and can be composed of *variables* and optional mathematical operators. Each variable inside an expression can be linked to a knowledge query. Figure 7 shows two examples of expressions.

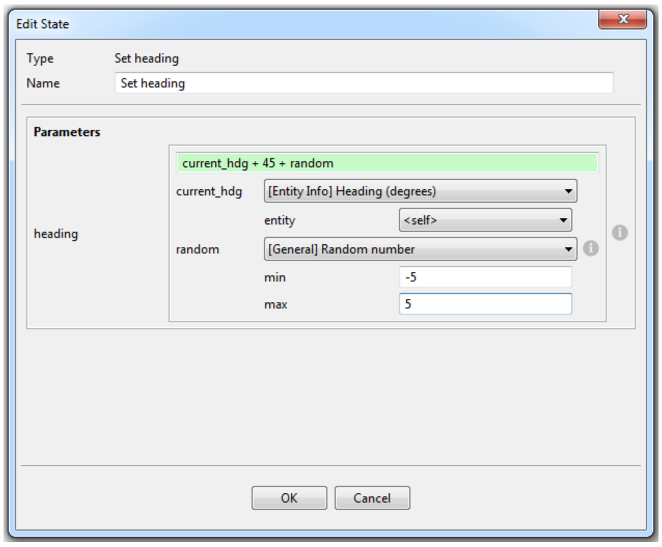

Edit Transition

Name	in weapon range
Source	Intercept
Destination	Radar illuminate target

☑ Condition — Quantify ☐

range < rOpt

range	[Entity Info] Range ground (NM)	
	source	<self>
	target	Target
rOpt	[Observation info] ROpt	
	shooter	<self>
	target	Target
	weapon	Alamo


Edit State

Type	Set heading
Name	Set heading

Parameters

heading: current_hdg + 45 + random

current_hdg	[Entity Info] Heading (degrees)	
	entity	<self>
random	[General] Random number	
	min	-5
	max	5

Figure 7: Expression in a decision-making condition (left) and action parameter (right)

In the left figure, an expression is shown that is used as a state transition condition in the decision-making module. It represents a condition of being *in weapon range* and becomes true when the range between the fighter jet and its current target becomes smaller than the optimal range (*Ropt* in fighter jet terminology) to launch a particular type of missile towards the target. In the right figure, an expression is used to specify a desired heading, namely a positive 45 degrees change from the current heading. Additionally by adjusting the new heading by a small random number we are able to simulate some noise in the fighter jet's calculations.

Blackboard

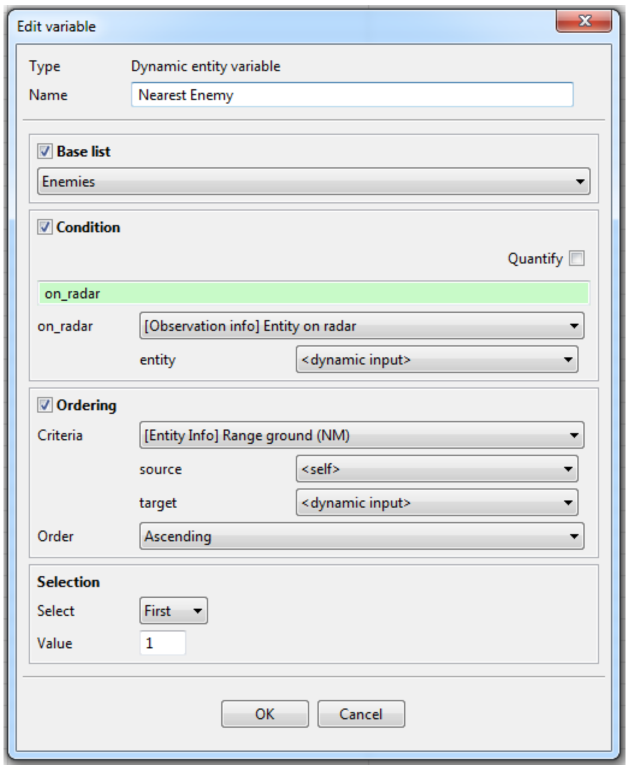
In the field of AI as well as in Smart Bandits, the term 'blackboard' refers to a knowledge base that can be globally accessed by agents for the storage and retrieval of knowledge. In comparison to the previously discussed knowledge, knowledge sources on a blackboard are introduced by the end-user himself. In Smart Bandits a blackboard system is available as a module. It allows end-users to store knowledge variables that represent numerical values or references to environment entity. Two types of knowledge variables are supported, namely *static* and *dynamic* variables.

A *static knowledge variable* can be defined by users to store persistent knowledge that can be set and retrieved at run-time. To give some examples: a numerical variable could be defined that represents the last time that a missile has been fired. It could be set whenever a *fire missile* action is performed, and queried when a course of action is desired after *x* amount of seconds after firing a missile. Furthermore, an entity variable could be introduced to represent a current enemy target to engage. It could be set when a fighter pilot has decided on a specific target to engage, and queried during the engagement to specify the object of interception or firing target.

A *dynamic knowledge variable* can be defined by users to represent rich composite knowledge that is evaluated continuously at run-time. The use of these variables is a powerful mechanism to acquire high-level situational awareness. These variables cannot be set manually at run-time but are configured at design-time to represent dynamic environment states. To give an example, one can define a reference to one or more entities that adhere to some specific condition which is evaluated at run-time. For instance, one can define entities such as ‘the current nearest friendly’, ‘current enemies detected on my radar’ or ‘current enemies within a certain range that are approaching me’. An example of a dynamic entity variable is shown in Figure 8.

As can be seen in the figure, a dynamic entity definition consists of *filtering*, *ordering* and *selection* criteria that can be specified. First, the filtering criterion filters all currently known entities (that an agent is aware of) using a *base list* and a *condition* that each known entity should adhere to. For example, a base list refers to another entity definition such as ‘all enemies’. A condition is defined by an expression that can reference knowledge queries, e.g. ‘all entities currently detected by radar’. After filtering, the *ordering* criterion allows one to order filtered entities using additional conditions, e.g. ‘order by distance’. Finally, the *selection* criterion allows one to filter the final list by selecting the first entity, a range of entities in the list or all entities. In conclusion, the example shown in the figure represents an entity definition that refers to ‘the closest enemy currently on radar’.

The advantage of dynamic knowledge variables is that it allows users to specify a priori concepts which are not known beforehand but can be resolved during the simulation based on an agent’s real-time situational awareness. Consequently such concepts can be used in decision-making or specifying action parameters.


Edit variable

Type Dynamic entity variable
Name Nearest Enemy

☑ **Base list**
Enemies

☑ **Condition**
Quantify ☐
on_radar
on_radar [Observation info] Entity on radar
entity <dynamic input>

☑ **Ordering**
Criteria [Entity Info] Range ground (NM)
source <self>
target <dynamic input>
Order Ascending

Selection
Select First
Value 1

Figure 8: Dynamic entity variable

3.6 Communication

Communication is an essential capability required by agents to coordinate their activities and exchange tactical information. Smart Bandits includes a communication mechanism that allows behavior models to interact. Two methods are supported both of which are described below.

Message Communication

Agents can send and receive messages to and from other agents using a common shared vocabulary. The vocabulary consists of user-defined messages that can be used in the decision-making module for sending and receiving. Sending of messages is known as a communication action and can be used just as any other action. Reception of messages can be detected within conditions to trigger transitions. Messages can be sent to one agent (e.g. a lead fighter sending a message to its wingman to initiate a certain tactic) or a group of agents (e.g. broadcasting to all friendlies). User-defined messages can also be parameterized with variables, making it possible to exchange more detailed information that is evaluated at run-time (e.g. a lead fighter that assigns a specific target to its wingman).

Sharing Situational Awareness

As an additional facility, a specialized communication action is offered which allows agents to communicate beliefs about an entity or group of entities all at once. For instance, an agent can send all beliefs it currently has about all its radar-contacts to its team-members. This facility can be used to simulate a form of shared situational awareness between team-members, similarly to what can be achieved by tactical data link network communication.

4 Future Directions

Smart Bandits originated out of a national research project with the goal to develop intelligent agents for tactical training for air-to-air engagements with simulation. Since then it has evolved into an operational platform to support the development of general-purpose intelligent agents. Smart Bandits continues to be used as a test-bed for new research projects related to training, modeling and simulation. It is also employed in a broader scope at NLR’s Battle Lab for CD&E purposes. NLR is continuously on the lookout for new developments that provide added value to the behavior modeling experience. Below we present the three R&D directions that we aim for in the near future.

Interoperability

The integration of Smart Bandits with a simulation platform requires the definition of an interface that specifies the *actions* an agent can use to control its physical entity as well as *observations* an agent receives from its physical entity (e.g. sensor data), giving it its situational awareness (see section 3.2). Such an interface has to be specified for different *types* of entity, such as a fighter aircraft, a submarine or tank. It is evident that the use of standards for such interfaces would be beneficial. Although standard interfaces could be defined geared towards a specific simulation platform (e.g. VBS), general standards for CGF platforms are currently non-existent. In the military simulation domain, development of standards to improve interoperability is an active field of research. E.g. consider HLA RPR-FOM for simulation state synchronization, MSDL for scenario specification or C-BML for information exchange between C2 systems (or C2SIM combining the former two). As the need for more agent-oriented behavior modeling for CGFs increases, with this comes the need for standard behavior interfaces for CGFs. This is an active line of research at NLR.

Machine Learning

Machine learning (ML) techniques can be employed to automatically generate behavior models for agents by having them learn behavior rules through interactions with the environment across a large amount of simulation runs. The use of ML for the creation of behavior models offers many challenges, e.g.: What metrics should be used for learning desired or optimal behavior? How can such ML models be validated? Can they be made transparent such that their internal decisions can be understood?

In one research project the use of a method called *Dynamic Scripting* is investigated, which is an evolutionary reinforcement learning algorithm. By using human-authored rules, it mitigates the common black-box problem of most ML solutions. We have performed a first validation step of machine-generated behavior models, by comparing the behavior that they produce to the behavior produced by man-made behavior models. The models were applied in pilot-in-the-loop simulations, after which SMEs reviewed the behavior of the CGFs along several metrics. The machine-generated behavior models scored equivalently to the man-made models on a majority of the metrics. This indicates that the quality of the generated behavior models already approaches the quality of man-made models, while they only require a fraction of the time to make.

A second research project focuses on an end-to-end reinforcement learning approach that combines symbolic and sub-symbolic approaches. Sub-symbolic approaches such as neural networks excel in learning patterns, but these patterns may not be obvious to humans that need to understand what is being learned. By intertwining the sub-symbolic approaches with a human-readable symbolic approach, we aim to find effective ways of controlling the machine learning process.

Natural Language Communication

The third line of research focuses on augmenting agents with natural human-like communicative abilities. This allows more natural human-agent interaction which is especially useful for training purposes. For instance, trainees would be able to interact with autonomous team-members to coordinate their activities and exchange information. Alternatively, training instructors would be able to control agents in real-time during a training. Besides topics of speech recognition, synthesis and natural language understanding, research challenges involve the integration of interaction models with an agent’s decision-making and memory models.

For more information about Smart Bandits, including a video of Smart Bandits in action, please visit our website:

http://www.nlr.org/capabilities/smart-bandits-air/

Scan the QR code (left) with your smartphone to be taken directly to the Smart Bandits website.

If you have any questions regarding Smart Bandits, such as (but not limited to):

- *I have a simulation package, will Smart Bandits connect to this package?*
- *I am developing new training simulations, how do I integrate Smart Bandits to improve the training value of the simulations?*
- or *How do I obtain a copy of Smart Bandits?*

Please do not hesitate to contact the Smart Bandits team directly at the following email address: smartbandits@nlr.nl