Designing Realistic Behaviors for a Military Training Simulation

Newfel MESSOUCI

Simulation is a one of the main tools for military training; it is used to confront trainees with complex situations that would be expensive and potentially dangerous to perform in the real world. In this context the modeling of realistic automated forces becomes essential.
This talk will focus on the behavior modeling process, from the analysis of the operational requirements and the military doctrine, to the actual implementation, and concludes with a description of the test and validation process. Participants will be presented with guidelines for building a maintainable library of behaviors that are both perceived and measured as realistic. MASA SWORD, a simulation for command post training used by many armies throughout the world will serve as an illustration for this presentation.

Introduction

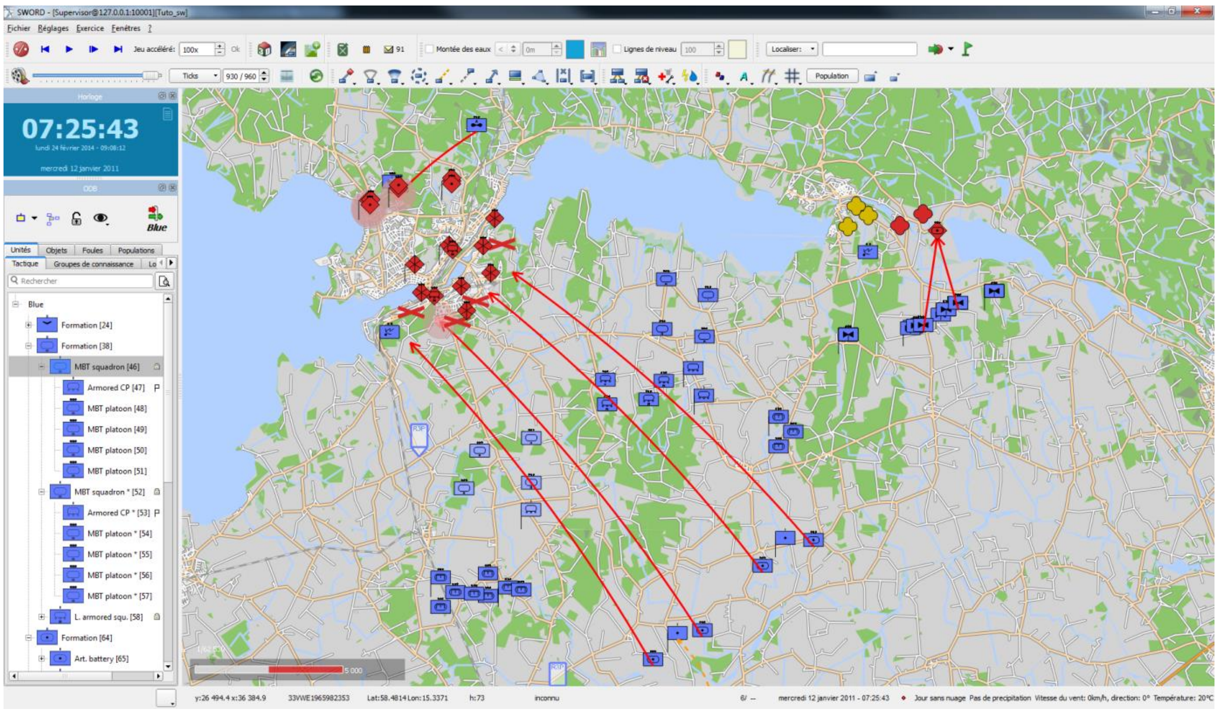
Simulations used for military training are classified in three categories:

- Live simulations: real people operating real systems. For instance, soldiers using actual vehicles on an open field to train for a specific real-life maneuver.
- Virtual simulations: real people operating simulated systems. A good example is a pilot training on a plane simulator.
- Constructive simulations: simulated people operating simulated systems. Mostly used to train command posts, these simulations allow the trainees to visualize, analyze and interact with large theaters of war with minimal overhead.

Constructive simulations are the central focus of our talk. Our main use case is to train high-ranking officers to make realtime decisions during military maneuvers. Had we used a live simulation, the cost would have been prohibitive: we would have needed to deploy thousands of soldiers and vehicles on an open field, with considerable logistic constraints. We therefore chose to use a constructive simulation.

During the exercise, the officers we want to train will be located somewhere resembling an actual command post, aware of the current situation on the simulated battlefield, but not interacting directly with the simulation (indeed, they may not even know a simulation is being used, although for practical reasons they usually do). Based on their current knowledge of the state of the battlefield, they call the operators who interact with the simulation to give them orders. The operators will then input the given orders in the simulation and report back to the officers with the results, who will therefore know the consequences of their decisions and can make new decisions based on their updated knowledge of the battlefield, and so on and so forth.

The problem we need to address is that a very simplistic constructive simulation needs one operator per agent in the simulation in order to control each of their individual actions. That is why we have a need to automate the entities and their decisions, and why we have a need for AI in our simulation. We will describe the behavior modeling process used for a constructive simulation called MASA SWORD, currently used by a wide range of armies across the world.



Behavior modeling process

1 - Military doctrine and user needs analysis

The behavior modeling process typically starts with a client asking us for a new mission required in the simulation, for example a behavior aiming to control a population, or to hold a position, etc. The first step is to read the military doctrine documents pertaining to the required behavior. These documents tend to be straightforward, usually consistent between different armies, but they're mostly guidelines and may not be enough for our needs. We need to go further and talk to military personnel.

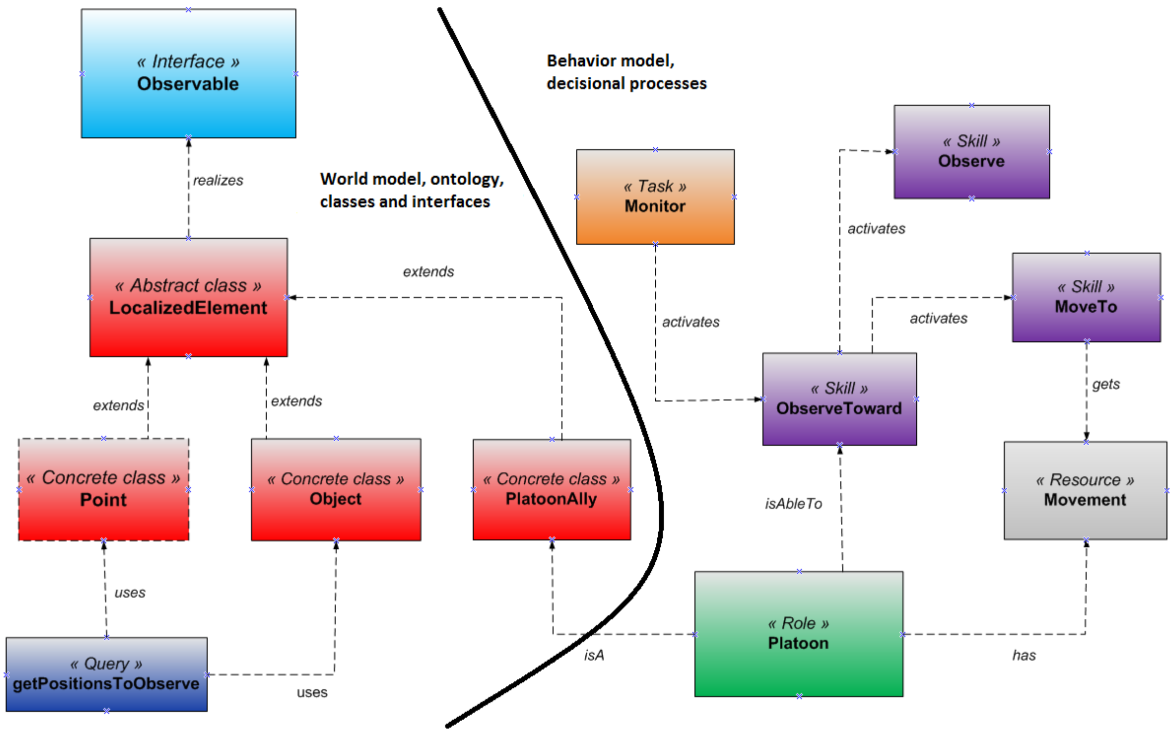
Interviewing doctrine experts is an essential part of user needs analysis, as they have invaluable insight into the behaviors they require in the simulation. However, other problems may start to arise: there may be a lack of consensus between officers over what a specific behavior should do, or they may go into too much detail over what the behavior should do, including detail ultimately irrelevant to the training. Finding the appropriate level of realism for the behaviors we model starts with extracting and understanding what is actually important to the trainees and the experts (even if they don't know it themselves).

This brings us to our main objective: to make realistic behaviors for our simulation. But what is realism according to the users? Operators need to see the behavior they expect when they give the mission to units, while trainees, who do not actually see the behavior but only hear about its effects, need to be satisfied that the feedback they receive is appropriate and makes sense, and ultimately satisfies their training needs. For these reasons, specifying the behaviors properly so that their effects match the training purposes of the simulation is paramount to ensure the product meets the client's expectations.

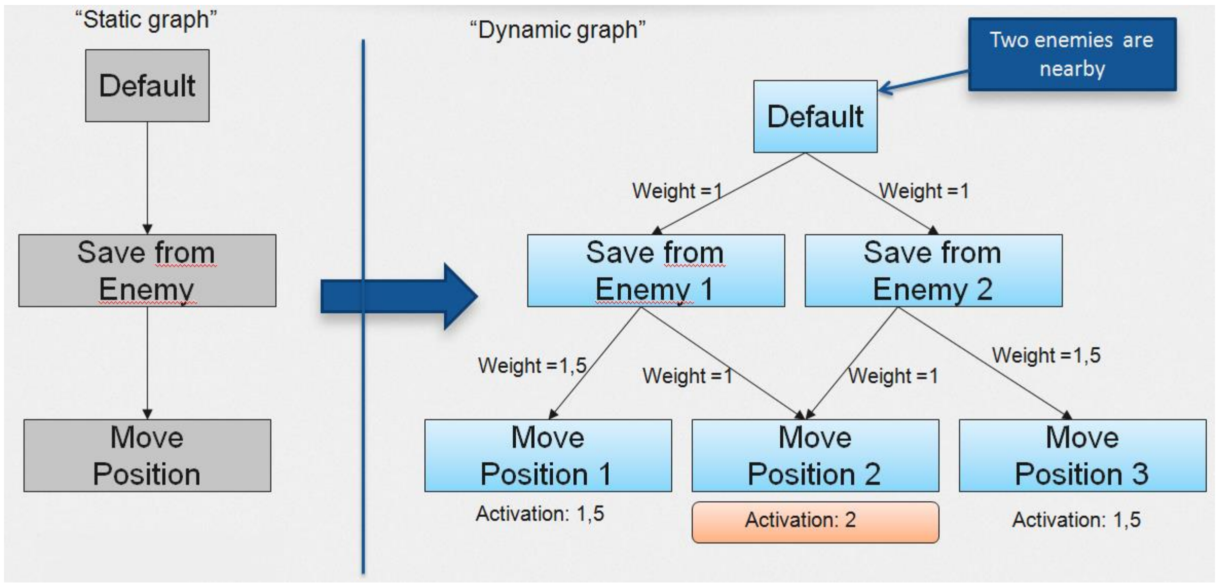
2 – Design and implementation

Now that the behaviors have been properly specified, they are implemented using our proprietary AI engine called Direct AI. This gives us access to a world description language allowing us to define an ontology (based on object-oriented programming principles), and this ontology is then used by our behaviors to query, change and analyze the current state of the simulation, ensuring that the agents are able to adapt to new situations as they arise. The engine uses a free-flow hierarchy to select actions, and our entire behavioral database is written in Lua (for the actual implementation) and in XML (for the declarative part). We edit our behavioral database with our custom IDE, an Eclipse plug-in.

Shown below is a small subset of our database. Behavioral and ontological parts are clearly indicated.



The entry point is the yellow box, the task called *Monitor*, corresponding to a single mission. This task activates the *ObserveToward* skill (a behavior), which in turn activates two lower levels skills: *Observe* and *MoveTo*. The *ObserveToward* skill makes the agent move towards a position, and once the position is reached, the agent observes the specified objective (such as an enemy, an agent, a direction). These skills are available to platoons (the green box), and all these behaviors use the world description (comprising concrete classes, abstract classes, and interfaces), in order to adapt to the current situation on the simulated battlefield. Our AI engine allows us to activate skills in sequences similar to the example above, but can also resolve conflicts between skills when they run in parallel.



For instance, let's assume an agent has a default behavior (i.e. a behavior that is always activated) that makes it retreat when faced with dangerous enemies. This behavior looks for safe positions, and the agent moves towards these positions. A static graph is defined in the behavior model: the default behavior activates a *Save from Enemy* skill, which in turn activates a *Move Position* skill in order to move to safe positions.

At runtime, the graph becomes dynamic as enemies appear near the agent: for every enemy from which protection is sought, an instance of the Save from Enemy skill is created. In our example, there are two enemies that represent a threat, so there are two instances of the Save from Enemy skill. The first instance of the skill computes two safe positions, and activates a movement skill towards each of these positions. The weight for each activation corresponds to the ability of the computed position to provide protection from a given enemy. For instance, the first one is very appropriate for protection against the first enemy, while second is less effective. The second protection skill also computes two positions, and ultimately the agent chooses the best compromise, meaning that it chooses the position that provides the best overall protection for all enemies.

In our example, the agent made a dynamic decision at runtime based on the current state of the simulation thanks to a static graph designed beforehand. Building these graphs requires an extensive library of atomic behaviors to serve as building blocks for our missions.

3 – Validation

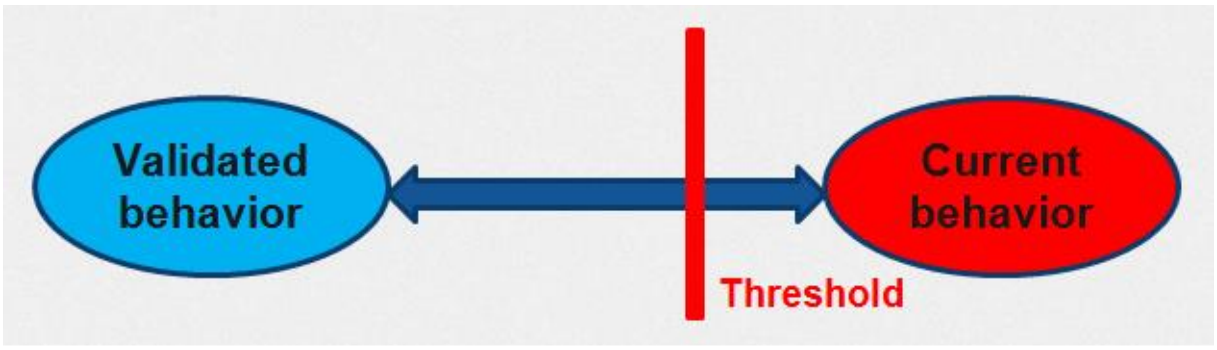
To validate the implemented behavior, multiple opinions are taken into account, and a lot of people have their say: the behavior modelers, the product manager, our operational counsel, the quality team and occasionally the clients themselves. Behavior validation is carried out on a full-scale scenario tailored to show off all the use cases of the behavior. In this way, we can make sure our new behavior suits the client's needs.

Once the behavior has been validated, we must guarantee backward compatibility with other behaviors, meaning that all new behaviors must not break previously delivered behaviors. To that end, we implement non-regression testing.

Following validation, we run the behavior in a scenario and record everything that happens. At a later stage, every time a new behavior is implemented, we start the scenario again with the new database, record everything, and then compare the two behaviors by computing a distance between them, which depends on several factors:

- The distance between the final positions of the agents
- The reports sent by the agents during missions
- Shots fired (for offensive behaviors)
- Specific actions performed, such as obstacle construction, decontamination, etc.

Once the distance has been computed, we compare it to a threshold of tolerance, and if the distance exceeds the threshold then this is deemed a regression, and the new behavior cannot be delivered in its current state.



Conclusion

Behavior modelers are the bridge between the military world and the simulation, and to ensure we design behaviors that the client will deem realistic, we must remain close to the end users in order to better understand their needs, and use our expertise in both simulation and military domains in order to measure what is truly important to them for their training purposes. Moreover, we need to determine the effects they expect from missions, because that is their main guideline when deciding whether a behavior is realistic or not. Having a keen understanding of how simulations are used by the military is therefore crucial.