Using Agent-Based Tactics Models to Control Virtual Actors in VBS3

(Demonstration)

Rick Evertsz, John Thangarajah, Nik Ambukovski
School of Computer Science and IT
RMIT University, Melbourne, Australia
{firstname.lastname@rmit.edu.au}

ABSTRACT

The computer-based simulation of military tactics has largely involved the use of platform-dependent scripting resulting in behaviour that is limited, and difficult to debug and reuse. The Tactics Development Framework (TDF) addresses these shortcomings by providing design-level support for tactics modelling. Its integration with VBS3 is reported here.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design—*methodologies*

Keywords

AOSE methodology; multiagent system; agent design models; cognitive modelling

1. INTRODUCTION

Military, immersive, computer-based training increasingly relies upon the realism of the tactical behaviour of *virtual actors* (computer-generated entities that represent human characters). Currently, simulations of tactical behaviour are scripted in painstaking detail, and this has three major shortcomings:

- The scripting languages are prescriptive and limited in scope, resulting in inflexible behaviour.
- The scripts are difficult to reuse, meaning that new behaviours have to be implemented from scratch.
- It is burdensome to transfer the tactics to other simulation platforms.

These factors limit the realism, usability and reusability of current tactical behaviour models. Reuse is also problematic because the models are only represented in terms of implementation (e.g. scripts); there is no direct mapping to high-level designs. Tactics models need to be realistic, easy to develop and use, and should be independent of the simulation platform. To address these shortcomings, we have developed TDF (Tactics Development Framework), a methodology and tool that supports the development and deployment of tactical behaviour models [1].

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

TDF is targeted at dynamic tactical domains that depend upon the tactician's ability to balance reactivity with proactivity. The ability to switch approaches or goals when the situation changes is fundamental, and is a good fit for BDI (Beliefs, Desires, Intentions) modelling [3]. Hence, we have adopted an AOSE (Agent Oriented Software Engineering) approach.

In this work we provide a demonstration of how high-level tactics designs, defined in the TDF tool [1], can be used to drive the behaviour of virtual actors in a state-of-the-art military simulation environment (VBS3), thereby overcoming the major shortcomings of platform-dependent, scripted approaches to tactics modelling.

2. TDF METHODOLOGY AND TOOL

TDF partitions tactics modelling into 3 main stages:

- **System Specification.** Identification of system-level artefacts, namely missions (key objectives and their properties), goals (hierarchies specifying how to achieve mission objectives), storylines (different ways the mission can play out), percepts (environmental input), actions (to perform on the environment), data (to store and access), actors (external entities) and roles (functionality required of the system).
- **Architectural Design.** Specification of the internals of the system, including the different agent types, the interactions between the agents (via protocols), and messages (sent between agents).
- **Detailed Design.** Definition of the internals of the agents, i.e. plan diagrams (diagrammatic representations of procedures), internal events (to trigger plans), messages sent and received, and data that is used by the agents.

TDF also encourages the definition of *tactics design patterns* – general-purpose tactical solutions that can be customised for related applications. The TDF tool is implemented as an extension of the Prometheus Design Tool (PDT) Eclipse plug-in [2]. It adds missions, tactics, plan diagrams and new goal control structures (asynchronous, concurrent, conditional and maintenance goals) to PDT. The TDF plugin has desirable features such as type safety, automatic propagation, batch image export, and code generation. These features play a significant role in ensuring that design artefacts created by the TDF tool are sound, and that the design maps to executable code.

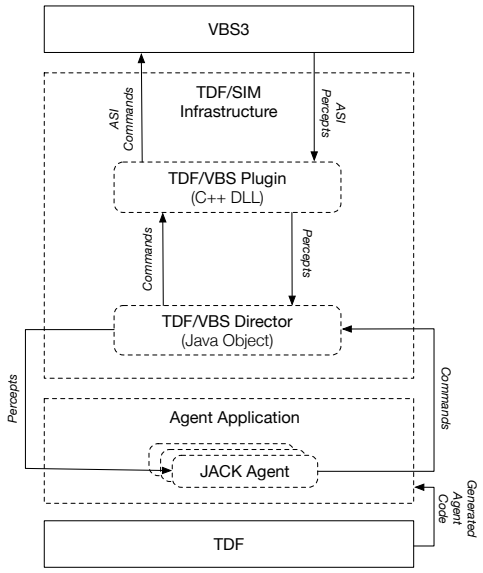

Figure 1: TDF/SIM Infrastructure

3. VBS3 INTEGRATION FRAMEWORK

Used by defence forces around the world, Virtual Battlespace (VBS) is the leading military 3D, photo-realistic virtual environment. Like other virtual environments that focus on visual realism, its inbuilt models of human behaviour are fairly rudimentary, and its scripting language is not well suited to the task of implementing dynamic and flexible tactics. With these shortcomings in mind, we have integrated TDF with VBS3 to allow the incorporation of richer models of human tactical decision making. Because TDF is a tactics design framework rather than a tactics execution environment, we have used its JACK [4] code generation capability to interface to the *TDF/SIM infrastructure* that we have created to link to VBS3. Figure 1, illustrates the components of our framework that facilitate the integration.

The TDF/SIM infrastructure consists of (i) TDF/VBS plugin, that communicates with VBS3; and (ii) TDF/VBS Director, that interfaces with the JACK agent(s) as shown in the figure.

Percepts are received from, and commands are sent to, VBS3 using its Application Scripting Interface (ASI), which provides access to VBS3's scripting language. Percepts are generated by VBS3 at the frame rate of the simulation (of the order of 50-100 frames per second). The TDF/SIM infrastructure reduces the perceptual load on the agents by filtering out any percept whose value has not changed since the previous frame.

4. EXAMPLE SCENARIO

An infantry L-ambush scenario provides a good demonstration of the combined capability of TDF and VBS3. In an L-ambush, the ambushers position themselves behind cover. The most effective course of action for the ambushees depends upon a number of factors, including ambusher proximity, availability of cover, and whether the ambushees are all in the "kill zone". These contextual variables are represented in TDF as `conditional goals`, and ultimately form

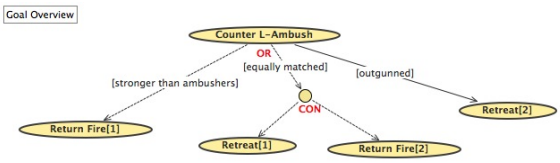

Figure 2: Goal Overview Showing Conditional Goals

the context conditions of the associated plans. In contrast to scripts, the tactical alternatives are clearly represented at a high level of abstraction in TDF. Figure 2 shows a goal overview diagram for countering an L-ambush.

5. DISCUSSION

We see the integration with VBS3 of a design-level tactics modelling tool, such as TDF, as an important step in the quest to build reusable libraries of tactics that can be used with different simulation platforms. The lack of a design-level tactics modelling methodology and tool has hindered the development of tactics libraries, because they are cumbersome to build and are tied to a single simulation platform.

Support by simulation platforms for external control needs to improve. Writing scripts to generate high-level percepts can be difficult and time consuming, and could be avoided if the simulation platform provided a greater variety of percepts through its API. There can also be delays in the platform's response to incoming commands, which in the case of VBS3 may be due to its underlying pathfinding process. This negatively impacts performance, and is particularly noticeable in training applications.

Greater independence from the particular simulation platform can be achieved in the future by developing an ontology that defines the percepts, actions, and the types of object available in the simulation environment.

ACKNOWLEDGEMENTS

We are grateful to the Defence Science and Technology Organisation (Australia), and the Defence Science Institute (Melbourne, Australia) for the invaluable support given to this project.

REFERENCES

[1] Rick Evertsz, John Thangarajah, Nitin Yadav, and Thanh Ly. Agent oriented modelling of tactical decision making. In Yolum, Weiss, Elkind, and Bordini, editors, *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, Istanbul, 2015.

[2] L. Padgham, J. Thangarajah, and M. Winikoff. Prometheus design tool. In *Proceedings of the 23rd AAAI Conference on AI*, pages 1882–1883, Chicago, USA, 2008. AAAI Press.

[3] A.S. Rao, M.P. Georgeff, et al. BDI agents: From theory to practice. In *Proceedings of the first ICMAS (95)*, pages 312–319. San Francisco, 1995.

[4] M. Winikoff. JACK intelligent agents: An industrial strength platform. *Multi-Agent Programming*, pages 175–193, 2005.