Tactics Development Framework

(Demonstration)

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

Thanh Ly
DSTO
Rockingham, Australia
thanh.ly@dsto.defence.gov.au

ABSTRACT

The Tactics Development Framework (TDF) is a tactics modelling application that extends the Prometheus Design Tool with tactics design patterns, plan diagrams, a mission concept, and richer goal hierarchies.

Categories and Subject Descriptors

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

General Terms

Design

Keywords

AOSE Methodology; Multiagent system; Agent design models

1. INTRODUCTION

In military domains, tactics lie at the heart of the computer simulation of combatant behaviour. Tactics are adversarial in nature and, taken together, specify the *means* of achieving the mission objective.

Over the last 20 years, our users' experience of modelling military tactics has revealed a recurrent theme: model reuse is problematic, particularly when a model is shared across team members. It is not unusual for a developer to implement a model from scratch rather than try to understand and reuse another's model.

Our user community is actively involved in the study of Undersea Warfare (USW), and has been building tactics models using a hybrid approach of UML and paper-based workflow diagrams. This approach does not scale well and leads to a shortfall in traceability between requirements, design and implementation, resulting in problems with model validation. This has led to a requirement for software engineering support of the tactics modelling process.

In the USW domain, information about an adversary's activities is difficult to acquire and involves a high degree of uncertainty. Assumptions are made that have to be revised as more data arrives, leading to the need to suspend the current plan and deal with a sudden change in the situation.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

The BDI (Beliefs, Desires, Intentions) paradigm [5] is well suited to modelling tactics in environments that are in a state of flux and involve uncertainty. Consequently, we have adapted the Prometheus BDI agent design methodology [4], extending it with explicit support for tactics modelling. We have developed TDF (Tactics Development Framework), which provides Agent Oriented Software Engineering support throughout the requirements and design phases to facilitate the modelling and implementation of tactics. TDF extends PDT (Prometheus Design Tool) with tactics design patterns, a high-level diagrammatic procedural representation, a mission definition, and richer goal structures. The link to implementation is supported through the generation of JACK [7] code stubs that the developer can fill out with more low-level detail.

2. DESCRIPTION OF TDF

In keeping with the Prometheus methodology, TDF partitions tactics modelling into 3 main stages:

- **System Specification.** Identification of system-level artefacts, including missions, goals, tactics design patterns, percepts/actions, actors, agents, roles and scenarios.
- **Architectural Design.** Specification of the internals of the system, including the different agent types (by grouping roles) and the interactions between the agents (via protocols).
- **Detailed Design.** Definition of the internals of the agents, including capabilities, plan diagrams, data, internal events and messages.

2.1 Missions

A mission is a goal-directed interaction between the system and its environment. It specifies the primary objective, secondary objectives, operational constraints, risks and opportunities. It includes a list of scenarios that can occur, data used and a natural language description of the overall mission.

2.2 Goal Structures

The goal structure shows the decomposition of the mission objectives into sub-goals. TDF supports and/or and concurrent goals, conditional goals, maintenance and preserve goals, and asynchronous goals. Additionally, goal siblings can be ordered or unordered.

Figure 1 shows a USW goal structure for handling an incoming torpedo. By default, the goals are tried in order

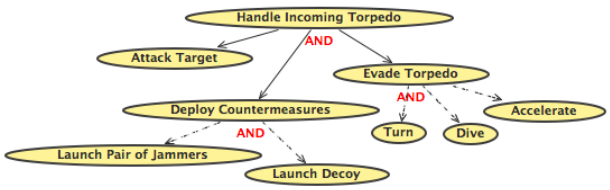

Figure 1: Goal Structures in TDF

from left to right. The submarine will first fire back at the attacker by trying to achieve the goal **Attack Target**. It will then **Deploy Countermeasures** and **Evade Torpedo**. The goals at the next level are unordered (signified by dotted lines), so for example, to **Evade Torpedo** the submarine can **Turn**, **Dive** and **Accelerate** in any order.

2.3 Tactics Design Patterns

TDF tactics design patterns are an idiomatic, BDI-influenced method of representing the intent of a tactic. They outline a tactic's main objective, triggering condition, problem description, solution description, invocation restrictions (**context**), outcomes, maintenance conditions, information required, information updated, goal structure, plan diagrams and the source of the information the tactic is based upon.

2.4 Plan Diagrams

Each plan diagram is diagrammatic pseudo-code, based upon UML Activity Diagrams, and represents part of an overall tactic. A typical tactic will include a set of plan diagrams. Iteration can be expressed as shown in Figure 2. The plan performs a **move to waypoint** action and waits until the next waypoint has been reached. It loops until the destination has been reached.

2.5 Scenarios, Roles, Percepts, Actions, Actors, Agents

Each scenario outlines part of how a mission could play out. Roles express functional groupings and are ultimately assigned to agents. The system interacts with actors, receiving percepts and performing actions.

2.6 Capabilities, Data, Events, Messages

Capabilities allow tactics to be structured into meaningful units that can be combined into agents. Tactics can involve the use and modification of data sources, the handling of events, and messaging between agents involved in executing a group tactic.

3. RELATED WORK

There are a number of AOSE tools including those based on the Tropos [2] and O-MaSE [3] methodologies. These tools focus on the decomposition of the system into functionally distinct components and how those components relate to one another. These tools are analogous to PDT, which TDF extends in a number of ways, most notably with tactics design patterns. Design patterns have been used for mobile agents [1], but they are quite similar to those found in object oriented design. An initial investigation of design patterns for human behaviour models has not progressed to the point of implementation [6].

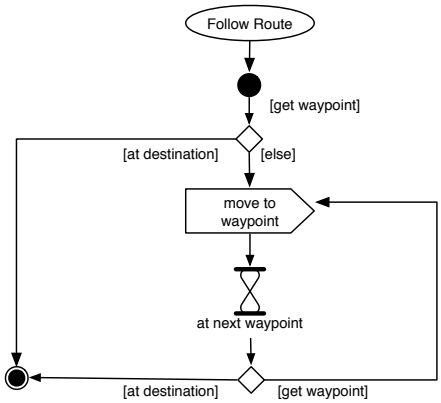

Figure 2: Waypoint Following Plan Diagram

4. CONCLUSIONS AND FUTURE WORK

TDF provides much needed design support for tactics modelling, and helps with reuse and sharing of models between developers. This is facilitated by the maintenance of the link between missions, goals, plan diagrams and implementation. This kind of support is essential for large, reusable tactics libraries.

TDF will benefit from extensions that perform consistency checking, for example, to identify goals that are not supported by a corresponding collection of plan diagrams. Management of large tactics libraries will benefit from the development of tactics ontologies with tool support within TDF. Another potential area for extension would be to address the knowledge acquisition phase, including tool support for interviewing Subject Matter Experts and linking such sources to existing or yet-to-be-defined design artefacts.

We acknowledge the support of the Defence Science and Technology Organisation, and the Defence Science Institute.

5. REFERENCES

[1] Yariv Aridor and Danny B. Lange. Agent design patterns: elements of agent application design. In *Proceedings of the second international conference on Autonomous agents*, pages 108–115. ACM, 1998.

[2] P. Bresciani, A. Perini, p. Giorgini, F. Giunchiglia, and Mylopoulos.J. Tropos: An agent oriented software development methodology. *AAMAS*, 8(3):203–236, 2004.

[3] S.A. DeLoach, W.H. Oyenan, J.C. Garcia-Ojeda, and J. Valenzuela. O-MaSE: A customizable approach to developing multiagent development processes, 2007.

[4] L. Padgham and M. Winikoff. *Developing intelligent agent systems: a practical guide*, volume 1. Wiley, 2004.

[5] 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.

[6] Glenn Taylor and Robert E. Wray. Behavior design patterns: Engineering human behavior models. In *Proceedings of the Behavior Representation in Modeling and Simulation Conference*, 2004.

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