Flexible and Reusable Tactical Behaviour Models for Combat Aircraft

Philip Kerbusch, Paul Eigeman
TNO Defence, Security and Safety
PO Box 96864
2509 JG The Hague
NETHERLANDS

philip.kerbusch@tno.nl, paul.eigeman@tno.nl

The Royal Netherlands Air Force (RNLAF) currently has the need to replace its fleet of F-16 combat aircraft due to aging. As a result of the introduction of a new aircraft, with advanced stealth capabilities, new weapons and sensors, and extensive network enabled capabilities the existing operational concept needs to be drastically changed. In order to support the development of new operational concepts and to perform operational analysis, quantitative studies in the form of constructive and virtual analysis (human-in-the-loop) are performed.

In order for constructive analysis to discover trends and generate statistically reliable results, a large number of experiments have to be conducted. This requires analysis- and simulation models. Supporting the RNLAF, TNO Defence, Security and Safety developed a generic aircraft simulation model. This simulation model includes both sophisticated simulation of hardware components like sensors and missiles, as well as simulation of the aircraft's tactical behaviour.

In this paper we describe how behaviour modelling techniques from the domain of (serious) games were used to develop a composable and flexible behaviour module in which the aircraft's tactical behaviour is modelled. This behaviour module is kept separate from the simulation suite in which the aircraft is modelled, ensuring the behaviour module can be reused in combination with different simulation suites and for other applications.

1.0 INTRODUCTION

The advantages of using simulation in the military domain are obvious. One can establish and repeatedly practice situations that are otherwise hard to achieve, for example due to limited availability of live systems and personnel. Another advantage of using simulation in the ability to investigate a platform or a system that is still in its conceptual development phase or design phase.

TNO Defence, Security and Safety over the past 12 years has been involved in a project that supports the Royal Netherlands Air Force (RNLAF) in replacing its fleet of F-16 combat aircraft. Due to operational, technical, and economical aging the F-16 fleet will have to be replaced by a new, more advanced combat aircraft. The introduction of such a new weapons platform, with advanced stealth capabilities, new sensors, and extensive network enabled capabilities results in drastic changes in the operational concept, compared to its predecessor. In order to support the RNLAF with the development of operational concepts (CONOPS) and to perform operational evaluation towards the RNLAF participation in the Operational Test & Evaluation (OT&E) phase, quantitative studies are performed. These studies are conducted by performing both *constructive* and *virtual analysis*, and will culminate –during OT&E participation- in life (-virtual-constructive) analysis. Constructive analysis entails conducting a large number of experiments in order to discover trends and generate statistically reliable results. These trends and results are then verified in several human-in-the-loop virtual analysis experiments.

In order to be able to conduct a large number of constructive analysis experiments, using for example Monte-Carlo simulation, analysis- and simulation models are required. For this reason TNO Defence,

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **OCT 2010**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Flexible and Reusable Tactical Behaviour Models for Combat Aircraft**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **TNO Defence, Security and Safety PO Box 96864 2509 JG The Hague NETHERLANDS**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADA564696. Human Modelling for Military Application (Applications militaires de la modelisation humaine). RTO-MP-HFM-202

14. ABSTRACT
The Royal Netherlands Air Force (RNLAF) currently has the need to replace its fleet of F-16 combat aircraft due to aging. As a result of the introduction of a new aircraft, with advanced stealth capabilities, new weapons and sensors, and extensive network enabled capabilities the existing operational concept needs to be drastically changed. In order to support the development of new operational concepts and to perform operational analysis, quantitative studies in the form of constructive and virtual analysis (human-in-the-loop) are performed. In order for constructive analysis to discover trends and generate statistically reliable results, a large number of experiments have to be conducted. This requires analysis- and simulation models. Supporting the RNLAF, TNO Defence, Security and Safety developed a generic aircraft simulation model. This simulation model includes both sophisticated simulation of hardware components like sensors and missiles, as well as simulation of the aircrafts tactical behaviour. In this paper we describe how behaviour modelling techniques from the domain of (serious) games were used to develop a composable and flexible behaviour module in which the aircrafts tactical behaviour is modelled. This behaviour module is kept separate from the simulation suite in which the aircraft is modelled, ensuring the behaviour module can be reused in combination with different simulation suites and for other applications.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**8**	

Safety and Security carries out a separate research program in which –amongst others- a generic aircraft simulation model is being developed, which includes sub-models for, amongst others, flight mechanics, electronic warfare, several types of radar, and missiles. In order to be able to perform constructive analysis, apposed to human-in-the-loop virtual analysis, a tactical behaviour model for combat aircraft is required as well. Other solutions that model tactical behaviour for combat aircraft exist, e.g. TALUS [1] and EADSIM [2]. However, these solutions proved insufficiently flexible with respect to how the tactical behaviour is modelled. Therefore we developed a tactical behaviour modelling module that can easily be adapted to suit the needs of the experiments. By keeping the behaviour module separate from the simulation suite in which the aircraft is modelled we ensure the behaviour module can be reused in combination with different simulation suites and for other applications.

In the remainder of this paper we will first elaborate on the tactical behaviour model of the combat aircraft, and the JROADS simulation suite that is used to model the weapons platform and to perform different types of analysis. Next, we will present our solution for effectively modelling the aircraft's tactical behaviour in a composable, flexible, and reusable manner. Finally, we will discuss and present our conclusions.

2.0 TACTICAL BEHAVIOUR MODEL

For the purpose of the research program at TNO the tactical behaviour of a combat aircraft is modelled by subject matter experts of TNO and the RNLAF. The model itself is classified, hence only part of the model is presented as an example in Figure 1:

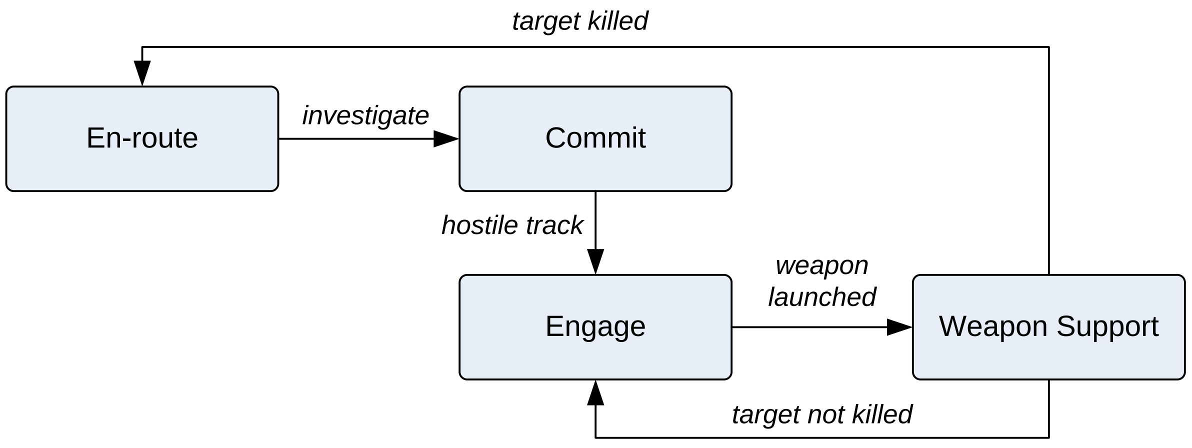


Figure 1: Part of the combat pilot tactical behaviour, modelled as a state diagram.

The tactical behaviour of the aircraft is modelled in an Object Oriented way. Figure 1 shows a so-called state diagram, where each state represents a single or series of actions. For example the state *Commit* would consist of choosing and executing an applicable (set of) manoeuvres towards the target until the target is within weapons range (or the action is aborted). The transitions between the states are triggered by events, e.g. the transition *hostile track* between states *Commit* and *Engage* occurs at a moment in time where a new hostile threat is detected and requires appropriate response.

As a result of the nature of the constructive analysis experiments to be conducted the model specifies the behaviour at a relatively high level. As an illustration of this consider the following example: A combat has fired a missile at its target. Since it is a guided missile, the missile needs to be actively supported (receive track-update messages). If at a certain moment a hostile incoming treat is detected a choice has to be made between starting an evasive manoeuvre whereby the missile is lost, or continuing to support the missile (for some time). This is a trade-off between the limited availability of missiles and the acceptable risk level that can be analysed by constructive analysis.

3.0 THE JROADS SIMULATION SUITE

In order to be able to model and simulate the combat aircraft at the desired level of fidelity a suitable simulation suite is required. For this purpose JROADS [3] is used. JROADS is a simulation environment in which weapons platforms can be modelled and scenario's containing multiple platforms (many-to-many) can be defined, simulated, and analyzed. JROADS is chosen as the simulation suite for modelling the combat aircraft due to its flexible and modular architecture and the number of 'off-the-shelf' components already available. As a result it is relatively easy to compose and refine an initial model of the combat aircraft.

JROADS originates from the '90s. At that time TNO developed a first version of JROADS in collaboration with the Royal Netherlands Navy (RNLM). In the following twenty years of incremental development JROADS was expanded and models of most modern air defence systems where added, both for the Royal Netherlands Army (RNLA) and the Royal Netherlands Air Force (RNLAF). Also extensive capabilities were added to support mission planning, post processing, and after-action review.

JROADS consists of several models for, among others, sensory systems, command and control (C2) systems, weapons and shooters. These models differ in their level of fidelity, e.g. ranging from cookie cutter sensors to high fidelity representations of specific sensors. Each platform modelled in JROADS consists of separate components that make up the full weapon system, as visualised in Figure 2.

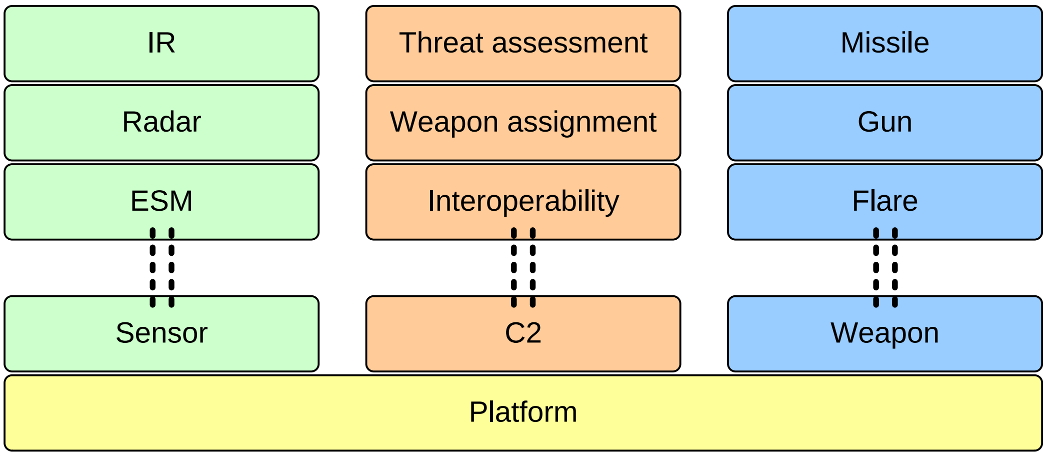


Figure 2: Composition of a weapons platform in JROADS. The platform is built by using components for sensors, command and control (C2) and weapons.

Depending on the requirements of the simulation, components can be replaced by components with a different level of fidelity. All components have well defined tasks, often split in hardware resembling components. For example a simple sensor system will be composed of a sensor component doing physical detections and creating sensor plots, and a tracker component that merges sensor plots into tracks. At the platform level the track is then used by the C2 component to decide what action should be taken. This could be the choosing and deployment of a weapon by a shooter.

In order to maintain the flexible and modular approach of JROADS, the different components that make up systems and eventually a platform communicate over well defined interfaces by means of a publish-

and-subscribe mechanism [4]. This implies that any component cannot directly change data owned by another component. Instead a message, in the form of an *order* or *notification*, is used, containing the data to be changed. This data is sent (published) to all components that are able to process this data (subscribers). Using the publish-and-subscribe mechanism has the advantage that it is easy to add or modify a component as long as it meets the interface specifications. This way loggers, for both debugging and analysis purposes, can also be tight to the model without interfering with the models themselves.

4.0 LUAFSM

When modelling the tactical behaviour of an aircraft JROADS exposes a shortcoming: components in JROADS are all modelled resembling hardware components, which are typically very predictable. Unlike a hardware component the tactical behaviour of an aircraft requires a high degree of dynamic and complex interaction, and thus requires a greater degree of flexibility: a real pilot would simultaneously interact with many of the aircraft's systems, and even the environment it is in. Although it is possible to create such a complex (hardcoded) component in JROADS, more fit for purpose solutions exist where the behaviour module is separated from the simulation suite [5].

A solution commonly used to model human behaviour in both entertainment and serious computer games and simulations is the use of a dedicated behaviour modelling module. These solutions provide various modelling paradigms tailored to human behaviour that differ with respect to complexity and the flexibility they provide. Examples range from the use of a lightweight scripting language like Python to full-blown agent frameworks that reason about their surroundings (e.g. Jadex [6]). It is unarguably best to choose a paradigm that is most fit for purpose given the requirements of the simulation and analysis experiments at hand.

4.1 Levels of Control

To create a fully functioning combat aircraft simulation model including tactical behaviour that can be used to conduct several types of constructive analysis, the tactical behaviour module has to be able to interact with the various hardware components. Thereby, the behaviour module should provide the end-user, e.g. TNO performing constructive analysis experiments, with the desired level of control. In the case of a combat aircraft manoeuvring for example, one could distinguish roughly three levels of control. In terms of level of abstraction from high to low: 1) defining the desired end position of the manoeuvre, without specifying how to reach it, 2) defining the desired end position and the required manoeuvres to reach it, and 3) defining the exact path the aircraft should follow to reach its end position. The desired level of control can be different every experiment, and even differ within various aspects of an experiment. This requires the behaviour module to be flexible in the possibilities it presents the end-user.

4.2 LuaFSM – JROADS

In Section 2.0 we presented the state diagram model for the tactical behaviour of the aircraft as constructed by the subject matter experts. In Section 3.0 we presented JROADS, and specifically how it uses the publish-and-subscribe mechanism to facilitate communication between separated components. On this basis we choose our behaviour modelling paradigm to be LuaFSM, a simple Finite State Machine [7] implemented in the dynamic scripting language Lua [8], created as part of this research. A FSM is chosen because it closely resembles how the aircraft's tactical behaviour is modelled in the state diagram, and it can easily be made event-driven, which fits the nature of the publish-and-subscribe mechanism. Lua is a lightweight yet powerful scripting language with extensible semantics that make it very flexible and thus suitable for this type of application. Lua is widely used within the simulation and entertainment gaming industry [9].

In order for the behaviour module to interact with the hardware components of the aircraft, the latter modelled in JROADS, an interface is required between the LuaFSM and JROADS. For this purpose the same publish-and-subscribe mechanism is utilized that JROADS uses internally. The behaviour module hooks into hardware component messages and is able to send these messages itself as well, as depicted in Figure 3.

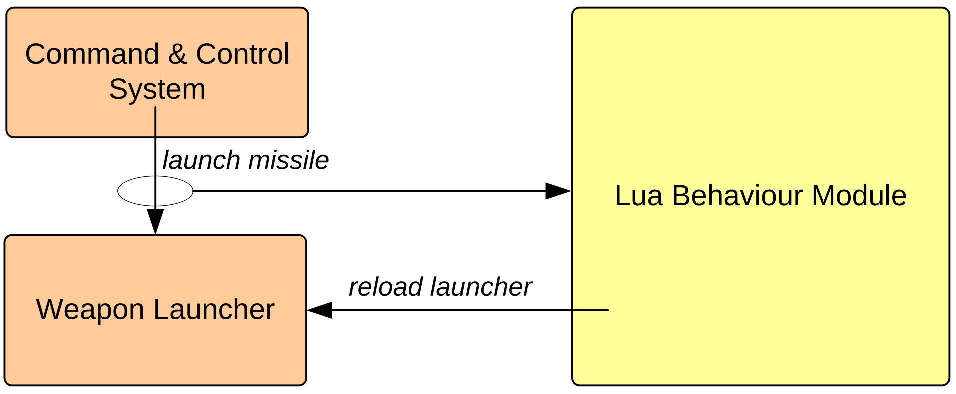


Figure 3: Example of the interaction between the Lua behaviour module and JROADS. The C2 system sends a message to a shooter, ordering it to launch a missile. This message is intercepted and passed to the behaviour module. The behaviour module sends a message back into the JROADS ordering the shooter to reload the launcher.

It is however more complex than depicted in Figure 3 to utilize the LuaFSM - JROADS interface is such a way that it is composable, reusable, and flexible in the level of control it presents to the end-user. Our solution is to use a configurable component in JROADS, the LuaAdapter, that on one side communicates with internal JROADS components via the built-in publish-and-subscribe mechanism, and on the other side to the behaviour module implemented in Lua. This architecture is depicted in Figure 4.

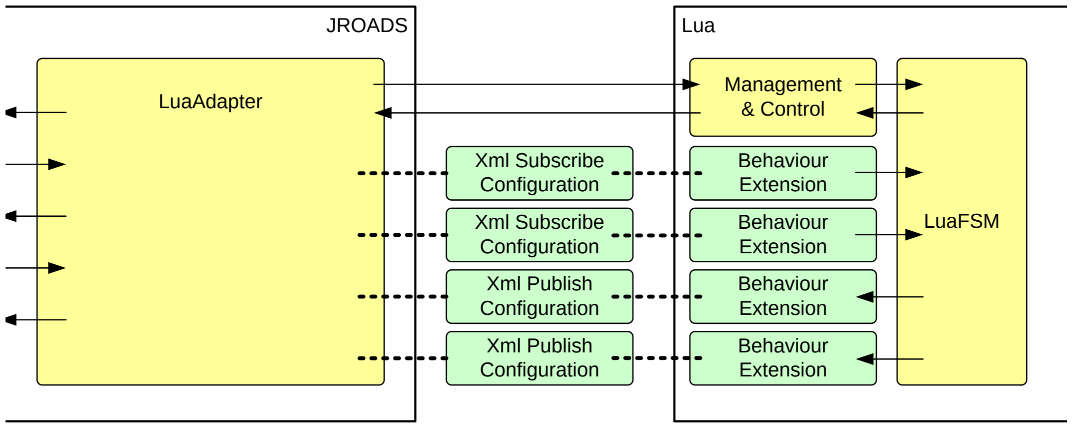
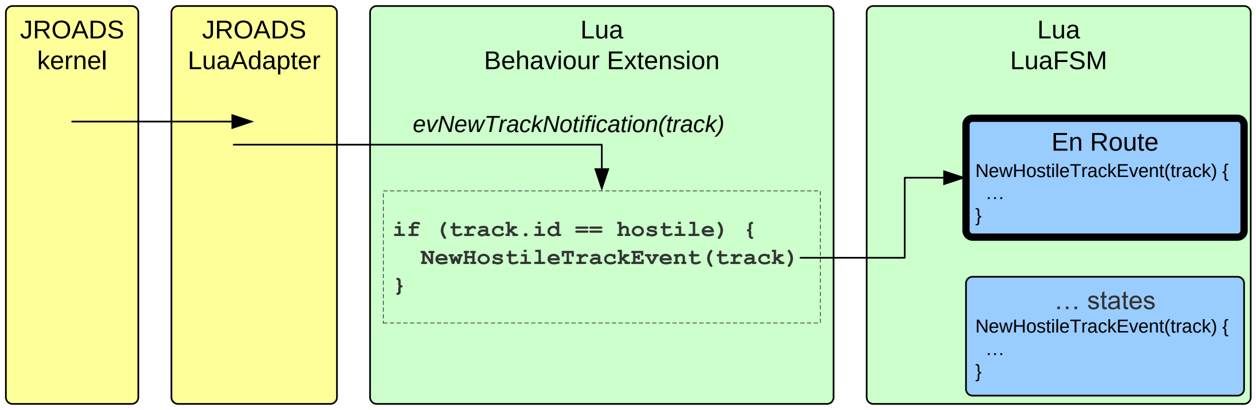


Figure 4: LuaFSM (R) and JROADS (L) interaction. The behaviour module is configured using pairs of Xml configurations and Behaviour Extensions.

From Figure 4 it can be observed that the configurable component in JROADS, the *LuaAdapter*, only directly communicates to the behaviour module in Lua for management and control issues (e.g. initialization). Thereby the LuaAdapter is neither explicitly aware of the specific behaviour it is modelling, nor the fact that a LuaFSM is used to model the behaviour. This has the advantages that 1) the LuaAdapter can easily be reused for other Lua applications, and 2) the LuaFSM can easily be replaced by a different behaviour modelling paradigm, e.g. behaviour trees [10].

The Xml *Publish* and *Subscribe* configurations, paired with a *Behaviour Extension*, both visible in Figure 4, play a central role in this architecture and give the behaviour module its flexibility and composability. The set of Xml Publish and Subscribe configurations instruct the LuaAdapter which notifications and orders it needs to subscribe to, and which notifications and orders it can send to the remainder of the JROADS components using the publish-and-subscribe mechanism. A Behaviour Extension contains the logic to deal with these notifications and orders, and it is these were the abstraction is made to achieve the desired level of control, before the behaviour module interacts with the LuaFSM. This is also shown in Figure 5; An *evNewTrackNotification*, which originates from the JROADS kernel, is processed in the Behaviour Extension. A new message, possibly at a different level op abstraction, is then send to the LuaFSM, where the current state processes the message.


JROADS kernel
JROADS LuaAdapter
Lua Behaviour Extension
evNewTrackNotification(track)
Lua LuaFSM
En Route
NewHostileTrackEvent(track) {
…
}
… states
NewHostileTrackEvent(track) {
…
}

```
if (track.id == hostile) {
  NewHostileTrackEvent(track)
}
```

Figure 5: Example of a Behaviour Extension dealing with levels of abstraction. A JROADS message *evNewTrackNotification* is generated by the JROADS kernel. This message reaches the Behaviour Extension via the LuaAdapter. A new message, at a different level of abstraction, is send to the LuaFSM, where the current state deals with the message.

By separating the concerns of the LuaAdapter and the LuaFSM, and being able to extend the behaviour module by stacking pairs of Behaviour Extensions and Xml configurations, we created a flexible, composable, and reusable behaviour module.

5.0 CONCLUSIONS

In this paper we presented our solution on how to include the tactical behaviour model of a combat aircraft in a simulation environment, in such a way that it is flexible, composable, re-usable, and provides the desired level of control to the end-user. In our case, the end-user is TNO Defence, Safety and Security conducting various constructive analysis experiments in order to support the development of new operational concepts (CONOPS) for the oncoming replacement of the current Royal Netherlands Air Force’s (RNLAF) F-16 combat aircraft.

For this research program TNO developed a generic combat aircraft model using the JROADS simulation environment. Advantages of the JROADS simulation environment are its modular architecture and the large number of components, e.g. sensors and weapons, already available. In order to support the performing of a large number of constructive analysis experiments (e.g. Monte-Carlo simulation) the tactical behaviour of the aircraft has to be modelled as well, preferably in a flexible and composable manner such that it can provide different levels of control and can easily be re-used for other applications.

In order to achieve this we created a Lua behaviour module that can easily be configured and extended stacking pairs of Behaviour Extensions and Xml Configurations. The actual implementation of the behaviour modelling paradigm is decoupled from the JROADS simulation environment, such that it can easily be replaced by a more advanced behaviour paradigm. Given the state diagram behaviour model created by subject matter experts we choose to use a finite state machine, LuaFSM, to model the behaviour.

A drawback encountered from the flexibility and dynamic nature of Lua is its lack of type-safety. As a result it is easy to oversee a typing error which generates unwanted results. In order to overcome this issue we created several run-time type-checking routines that evaluate the Lua code, especially the parts where the end-user is involved.

A drawback encountered from using the finite state machine paradigm to model the tactical behaviour is that it does not include a form of memory. This results in situations occurring where for example the pilot evades a SAM (Surface-to-Air Missile) site, later returns to its route, encounters the same SAM site, evades, returns to its route, and so on. By using a more advanced behaviour modelling paradigm that can deal with temporal information this could be resolved.

Current results show that the solution is flexible and able to provide the different levels of control desired by the end-user. The solution has thus far been used in several other occasions in conjunction with the JROADS simulation environment, for example in the naval domain for modelling the responses of 'suspect' ships targeted by non-lethal weapons. Future research may show that the provided solution can also be used for other types of application, for example intelligent real-time scenario generation.

[1] TALUS (1999). TALUS – an object oriented air combat simulation. *Proceedings of the 1999 Winter Simulation Conference*, *2*, 1160-1167.

[2] EADSIM (2010). Extended Air Defense Simulation (EADSIM). Retrieved from the web August 9, 2010. http://www.eadsim.com/

[3] JROADS (2010). Retrieved from the web August 9, 2010. http://www.jroads.com

[4] Birman, K. & Joseph, T. (1987). Exploiting virtual synchrony in distributed systems. *ACM SIGOPS Operating Systems Review*, *2*, 123-138.

[5] de Kraker, K.J., Kerbusch, P., and Borgers, E. (2009). Re-usable behavior specifications for tactical doctrine. *Proceedings of the 18th Conference on Behavior Representation in Modeling and Simulation*, 15-22.

[6] Jadex (2010). Retrieved from the web August 9, 2010, http://jadex-agents.informatik.uni-hamburg.de

[7] Fu, D., Houlette, R. (2004). The Ultimate Guide to FSMs in Games. *AI Game Programming Wisdom 2*, 283-302.

[8] Lua (2010). The programming language Lua. Retrieved from the web August 9, 2010, http://www.lua.org/

[9] Lua Video Games (2010). Retrieved from the web August 9, 2010, http://en.wikipedia.org/wiki/Category:Lua-scripted_video_games

[10] Champandard, A.J. (2007). Understanding Behavior Trees. Retrieved from the web August 9, 2010, http://aigamedev.com/open/articles/bt-overview/