CoJACK – Achieving Principled Behaviour Variation in a Moderated Cognitive Architecture

Rick Evertsz
AOS Group
Level 1, 594 Elizabeth Street, Melbourne, Victoria 3000, Australia
rick.evertsz@aosgrp.com

Frank E. Ritter
College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802
frank.ritter@psu.edu

Paolo Busetta,
Matteo Pedrotti
AOS Group
Wellington House, East Road, Cambridge, CB1 1BH, UK
paolo.busetta@aosgrp.com
matteo.pedrotti@aosgrp.com

Jennifer L. Bittner
Department of Psychology
The Pennsylvania State University, University Park, PA 16802
jlb503@psu.edu

Keywords:
Modelling and Simulation, Cognitive Architecture, Agents, CGF.

ABSTRACT: *We describe the CoJACK cognitive architecture. It is designed to provide an intuitive, high level behaviour representation language that exhibits variability across individuals and across time for use in synthetic environments. The included moderator layer enables modelling of physiological factors and affect. We report initial results of an investigation of variability and time-based moderation in the dTank simulation. The results show that moderation leads to interesting effects and that variability is generated in a principled and repeatable fashion.*

1. Introduction

Simulation is now an indispensable part of the military environment, and is used in many areas including training, tactics development, mission rehearsal, course of action analysis and hardware acquisition. Applications typically require the inclusion of simulated human entities because humans are an essential part of most operations (Pew & Mavor, 1998, 2007). However, in contrast to hardware such as aircraft, tanks and weapons systems, even highly-trained humans can vary significantly in their response to a given situation. Although the inherent variability of humans has been widely recognized, Semi-Autonomous Forces (SAF) have, for various reasons, tended to neglect this phenomenon. One reason for this is the difficulty of modelling the depth and breadth of human behaviour. In addition, early studies focused on obtaining predictable and repeatable results, for example, in basic training scenarios.

Despite their consistency, inflexible human behaviour models can lead to a false sense of confidence in the predicted outcomes of more advanced scenarios. This predictability may produce results that do not reflect reality (Poncelin de Raucourt, 1997) and can lead to poor training outcomes.

In fact, modelling the variance in human behaviour is not the antithesis of predictability or repeatability. By incorporating variability, we strive to predict both the *potential* for variation, as well as the *range* of that variation, not to create models that intrinsically behave differently each time they are run.

Our approach yields models that are deterministic; given the same initial state and time course of events in the Synthetic Environment (SE), the models will behave in precisely the same way every time. If, however, one alters the parameters of the underlying cognitive architecture (including a random seed), the models will generate different behaviour. These parameters can be altered manually through the GUI or automatically via moderators that are environmentally driven (e.g. temperature or precipitation) as well as the simulated human entity's physical or affective state (e.g. fatigue or fear).

CoJACK™ (Norling & Ritter, 2004) is a novel cognitive architecture, based on JACK™ (Busetta, Rönnquist, Hodgson & Lucas, 1999), which predicts how behaviour varies as a function of the changes to the architecture's parameters. It supports the definition of **moderators** that modulate these parameters and thereby predicts, in a principled way, how behaviour varies as a result of physiological and affective factors. CoJACK's novel contribution to the field lies in its synthesis of an easy-to-use BDI representation with a cognitive architecture. In contrast to cognitive architectures that break tasks up into very small steps (cf. ACT-R (Anderson, 2007)), CoJACK combines a high-level plan representation with sub-symbolic equations that influence computation without obscuring that high-level viewpoint.

After reviewing current behaviour modelling tools, we describe our approach with CoJACK and its current implementation. We also describe some preliminary experiments to explore variance in its behaviour, and the lessons we have learned so far.

2. Current Behaviour Modelling Tools

The representation of human behaviour in simulation began with attempts to model decision-making using conventional procedural software languages, such as Fortran. Although simple behaviours independent of context may be represented in this way, the code rapidly becomes complex and difficult to extend or maintain.

To overcome this problem, SAFs provided the ability to script behaviour. These scripts, built prior to the scenario's running, govern the behaviour of the entities, determining their movements and actions. However, because they are scripts, they are inflexible and thus a poor vehicle for modelling variation.

In an effort to provide more intelligent behaviour than is possible with scripts, researchers created models using Artificial Intelligence (AI) languages like Soar (Jones et al., 1999) and JACK. This resulted in much more realistic decision-making, but did not address the issue of individual variability (other than that which results from differences in each agent's knowledge). Consequently, variation has to be explicitly programmed in these models – an ad hoc and time-consuming process.

ACT-R (Anderson, 2007) is a cognitive architecture with sub-symbolic parameters that predict variation in cognition. These include noise parameters that can be increased to produce further variation. ACT-R was developed in the context of small laboratory tasks used to test psychological theories. The programming primitives in ACT-R (production rules) are fine-grained and represent cognitive operations of the order of 50 ms. This can make ACT-R models tedious to build, and potentially difficult for Subject Matter Experts (SMEs) to understand, and therefore less suitable for building models of human behaviour for use in simulation.

ACT-R is currently primarily concerned with cognition; but there are other architectures that address sources of variability, such as physiology and emotion. For example, PSI is a relatively new cognitive architecture that is intended to integrate cognition, emotion and motivational factors (Bartl & Dörner, 1998; Bach, 2007). PMFServ (Silverman, 2004) was designed to provide performance-moderating functions but has grown into a more complete architecture for representing affective states.

Gratch and Marsalla (Gratch & Marsella, 2004) provide a model of task appraisal and its effect on performance. Their model can be hooked up to simulations to generate behaviour, but they do not include other moderators. Hudlicka and Plautz (Hudlicka & Pfautz, 2002) provide a more general model, and provide the ability to work with arbitrary simulations. IMPRINT (Booher & Minninger, 2003) includes more moderators, but IMPRINT does not generate behaviour or interact with external simulations (but see (Lebiere et al., 2002), for work towards this end).

These architectures have various strengths and weaknesses, but none were designed to provide an intuitive, SME-friendly representation based on a moderated cognitive architecture. Our goal in developing CoJACK was to provide developers a high-level intuitive representation that is both underpinned by a principled cognitive architecture and capable of being moderated by physiological and affective factors. CoJACK adds principled variation to behaviour models based upon properties found in the human cognitive system. It also predicts how moderating factors such as fatigue and emotion affect the cognitive architecture. CoJACK achieves this without compromising JACK's high-level, intuitive tactics representation.

3. Design Approach

To understand CoJACK, it is necessary to have some basic knowledge of JACK's structure.

3.1 JACK

JACK is based on the Belief/Desire/Intention (BDI) model, a paradigm that explicates rational decision-making about actions. Drawing on Bratman's (Bratman, 1987) work on situated rational agents, this approach has been applied to a wide range of problems, including fault diagnosis for Space Shuttle missions (Georgeff & Ingrand, 1989). The BDI paradigm was developed in response to a perceived problem with existing AI approaches to the planning problem. Agents are typically situated in a dynamic environment and must constantly review their goals and activities. They should also be aware of the resource-bounded nature of their reasoning. Earlier AI approaches to planning only addressed the offline planning problem – how to achieve the goal given infinite time and resources. They failed to address the temporal pressures that apply when trying to achieve goals within a fluctuating environment that presents a multitude of interacting, conflicting and changing opportunities.

From the perspective of a cognitive architecture, the key programming constructs of JACK are:

- Events – are the central motivating factor in agents. Events are generated in response to external stimuli or as a result of internal agent computation.

- Plans – are procedures that define how to respond to events. When an event is generated, JACK computes the set of plans that are applicable to the event. These plans are subjected to a process of deliberation, where the agent selects the plan that will form its next intention. Plans have a body that defines the steps to be executed in response to the event. Non-deterministic choice allows the agent to try alternative plans to achieve the goal.

- Beliefsets – are used to represent the agent's declarative beliefs in a first order, tuple-based relational form. Beliefsets are analogous to the Working Memory (WM) of a production system.

- Intentions – are the currently active plan instantiations, i.e. the plan instances that the agent is committed to. Plans are abstract entities that describe potential patterns of thought and action. A plan becomes an intention when the agent instantiates it with symbolic references to concrete entities in the environment, and commits to its execution.

JACK represents and executes tactics in a manner that maps well to SMEs' introspections about their own reasoning processes. The tactics representation includes a front-end that allows analysts to specify tactics graphically at a high-level without having to worry about low-level detail. The graphical representation is extremely useful for visualising the logical structure of tactics and for discussing them with SMEs. Therefore, a key design goal for CoJACK was to retain JACK's high-level representation and ease-of-use.

3.2 CoJACK Extends JACK with Timing, Errors, and Moderators

CoJACK augments JACK with a set of cognitive architectural constraints and parameters, as well as a moderator layer. Based upon cognitive parameter values, the architectural constraints add latency to the current intention's reasoning steps and to memory access. CoJACK also affects the choice of beliefs retrieved in response to a memory access attempt; this includes effects such as failure to retrieve a matching belief, retrieval of a belief that only partially matches, and retrieval of an alternative matching belief (i.e. not the one that JACK would have chosen first). A similar mechanism affects the selection of the next intention to execute. Thus the agent can choose an unanticipated intention or even fail to retrieve one of its current intentions. The cognitive parameters can be moderated at runtime, leading to further variation in behaviour. For example, a caffeine moderator could be added that decreases the time taken to perform reasoning steps, leading to shorter response times.

3.3 Key CoJACK Assumptions and Features

The goal for CoJACK was to develop a cognitive architecture that predicts the timings and errors in human performance without compromising the usability of the BDI representation. We start with the assumption that humans share a common cognitive architecture and physiology, and that variation results from individual differences in knowledge and the values of the architecture's and physiology's parameters. CoJACK's cognitive parameters are taken from ACT-R and adapted for a BDI architecture. ACT-R's parameters have been experimentally validated across a wide range of tasks. We have found that they are an excellent starting point for a new architecture such as CoJACK. Cognition can be modulated by variations in physiology (as represented by the moderator layer; see next section).

One way to view cognitive architectures is to classify them in terms of the constraints they impose upon cognition models. Howes and Young (Howes & Young, 1997) point out that cognitive architectures consist of soft and hard constraints. Soft constraints can be overridden whereas hard constraints enforce particular properties on the models, with no way to circumvent them within the architecture. From this perspective, CoJACK is intended to provide hard constraints (because they are provided automatically to a JACK model), but because of the Java implementation and the ability to extend JACK and CoJACK, the constraints are somewhat soft. This approach offers considerable flexibility because the degree of cognitive modelling sophistication and consequent behavioural variance can be altered to suit the goals of the simulation study.

CoJACK addresses the following five key aspects of cognition:

Limited access to procedural and declarative memory – Working Memory (WM) refers to the currently active subset of the human cognitive system, in some sense, the contents of consciousness. WM is the mechanism of human cognition that maintains information during processing. It is limited in capacity as evidenced by decreased performance in tasks that require many temporary items to be held in memory. Although the term WM is usually restricted to declarative memory, it should also encompass the dynamic aspects of procedural memory. CoJACK incorporates WM access limits (termed errors of omission) through an activation-level mechanism that is similar to the ACT-R approach. In contrast to its support for declarative memory, ACT-R does not include a theory describing how recency and decay apply to dynamic procedural memory. Unlike ACT-R, this is a real issue for CoJACK because it represents procedures as plans rather than productions. A plan typically has a number of steps and when a CoJACK agent forms an intention, that intention is held in a dynamic memory buffer that enables the agent to step through the procedure. CoJACK can maintain a number of such intentions (determined by sub-symbolic activation-level equations), and switches its focus to the most active intention. This allows it to work on a number of tasks concurrently, much as a short-order cook manages multiple dishes (Kirlik, 2006).

Error-prone retrieval from procedural and declarative memory – In addition to failing to access memory elements, errors can occur when the *wrong* memory item is retrieved. A partial-matching mechanism (Anderson & Lebiere, 1998) accounts for some of these errors (termed errors of commission). CoJACK adopts a similar approach.

Cognition takes time – One of the major limitations of most AI-based models is that they fail to represent the time taken to think and act. If, for example, the granularity of the simulation is 1 second, a model can, in theory, and often in practice, run through a series of decision-making steps in 1 (simulated or real) second that would take a human half an hour. CoJACK addresses this problem by computing the time of reasoning steps (based on its moderated parameter set).

Limited focus of attention – Resource limitation is a key property of human cognition. The allocation of this limited computational resource is generally referred to as **attention**. In CoJACK, the modelling of attention takes account of the following:

How an agent deliberately focuses its attention in a particular direction (e.g. focusing on an important goal while ignoring distracting environmental stimuli.)

How factors like caffeine or fatigue moderate an agent's attention.

How the properties of memory affect attention. For example, WM elements have a limited life span. If a WM element is essential to the current task, the agent must counteract its natural decay through some form of rehearsal. This rehearsal process consumes attentional resources.

Cognition can be moderated – Human behaviour can be modified (moderated) by a range of factors, including temporal, environmental, physiological and internal factors. Moderators can influence entity behaviour directly – for example, caffeine typically provides a 10% faster reaction time on a simple reaction time task. Moderators can also influence lower-level mechanisms that give rise to the changed performance – for example, caffeine reduces reaction time and increases the ability to focus, allowing performance on a vigilance task to remain virtually sustained at its original level instead of decreasing over the span of an hour (for some subpopulations) (Boff & Lincoln, 1988). By simulating the effects of moderators on underlying mechanisms, it is possible to predict behaviour variation that will occur for a task that has not yet been studied closely. For example, if the effects of caffeine on the low-level aspects of cognition and the body are well understood, but the effects of caffeine on, say, radar operators had not been specifically studied, it would be possible to provide at least initial predictions of caffeine's effects on the behaviour of radar operators based on knowing their cognitive mechanisms and the knowledge necessary to perform the task.

4. CoJACK Details

Because JACK was intended for use in resource-limited environments (such as mobile autonomous systems), it is lightweight and efficient. An important goal for CoJACK was to provide an environment in which the user can easily enable/disable the cognitive architecture. When disabled, the agent runs as a normal JACK agent. When enabled, the agent's behaviour is modulated by the cognitive architecture. Therefore, it was important that CoJACK not modify the syntax of JACK. Furthermore, the CoJACK extensions were implemented so as to not affect the performance of normal JACK models. This was achieved by having the CoJACK extensions added at compile time if enabled for the agent in question.

4.1 Sub-symbolic Properties

Models in CoJACK are largely built at the symbolic level (specifying the plans, events and beliefs handled by an agent). However, the final behaviour of a CoJACK model depends on the sub-symbolic properties of the architecture.

In CoJACK, beliefs and intentions are subject to sub-symbolic effects. To unify their treatment, they are collectively termed **chunks** in CoJACK. A belief is a declarative memory chunk and intentions are procedural memory chunks. From the point of view of Working Memory (WM), a chunk is a single item[1].

Activation level is a key aspect of the sub-symbolic properties of CoJACK. Chunks have an activation level that changes over time as the chunk is used. This activation level is one of the main influences on the likelihood that a chunk will be retrieved, and how long that retrieval will take. Given a set of competing chunks (i.e. ones that match the retrieval specification), CoJACK will select the most activated chunk, subject to the chunk being above the retrieval threshold. Recall that CoJACK events govern the agent's computational flow (i.e. internal and external events produce intentions which then drive behaviour). When an event is processed, it acts as an activation source for chunks. Event activation can be moderated to support goals with varying "salience" (e.g. "Stay alive" might have a higher activation than the goal "Watch TV"). This provides a straightforward method for the developer to focus the agent's attention on pertinent events (e.g. those that are threatening to the agent).

CoJACK has over 30 cognitive parameters that govern a wide range of sub-symbolic properties including: chunk activation, the time taken to retrieve or modify memory elements, and various noise factors. Individual differences between agents can be represented by supplying differing initialisation parameter values. Moderators can also further affect these cognitive parameters at runtime, for example, a "fatigue" moderator can be added that increases the time taken to retrieve or modify memory elements.

4.2 Moderator Representation

Each moderator is represented as a mathematical function denoting its input/output mapping. Moderators can also have internal reservoirs and decay functions that determine how the reservoir level varies over time (e.g. to model the rate caffeine is excreted). Composite moderators can be created that represent the interaction between one or more moderators (e.g. the interaction between caffeine and alcohol, and how that affects reaction time). The inputs to a moderator can be (i) events from the simulation environment, (ii) internal events (e.g. the timing and amount of caffeine dosage where the simulation environment does not provide such data), and (iii) outputs from other moderators.

4.3 CoJACK in Synthetic Environments

Figure 1 is a conceptual model of how a CoJACK agent fits into an SE. Tactics are represented using JACK's standard graphical BDI representation, but their performance is modulated by the underlying CoJACK architecture. Thus, whereas a JACK agent could instantly memorise a lengthy Rules of Engagement card, a CoJACK agent would exhibit Working Memory limitations. The figure shows two moderators: fatigue and caffeine. These reside in the Moderator Layer and feed into the cognitive architecture's parameter set, leading to moderated cognition (e.g. hampered memory retrieval in the case of fatigue). The CoJACK agent has an

[1] Even if an intention has, say, 50 steps, it will be treated as a single WM item. Thus, if the modeler creates a plan with 50 steps, it signifies that the plan is "compiled" and therefore its WM burden is low.

embodiment that represents aspects of its physical body. Sensory and physiological information from this embodiment feeds into the moderators (e.g. caffeine intake tops up the reservoir in the caffeine moderator). The agent embodiment overlaps to some extent with the embodiment in the SAF itself. This boundary is shown as a dotted line because extent of this overlap varies depending on the SAF in question. The SAF provides sensory information (perception) and effects the actions specified by the agent. Additionally, the agent embodiment can take non-sensory input from the SAF (e.g. thermal energy based on the ambient temperature).

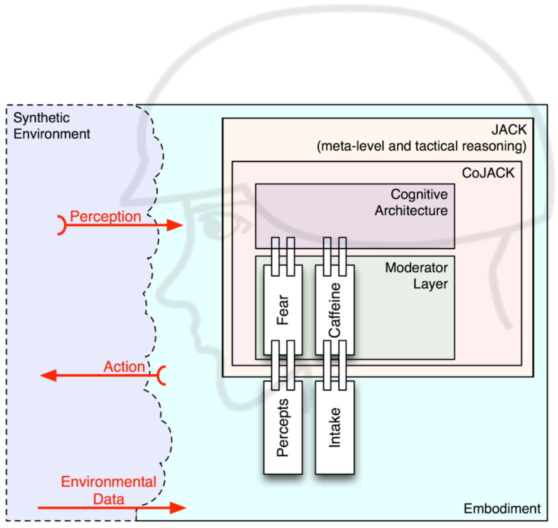
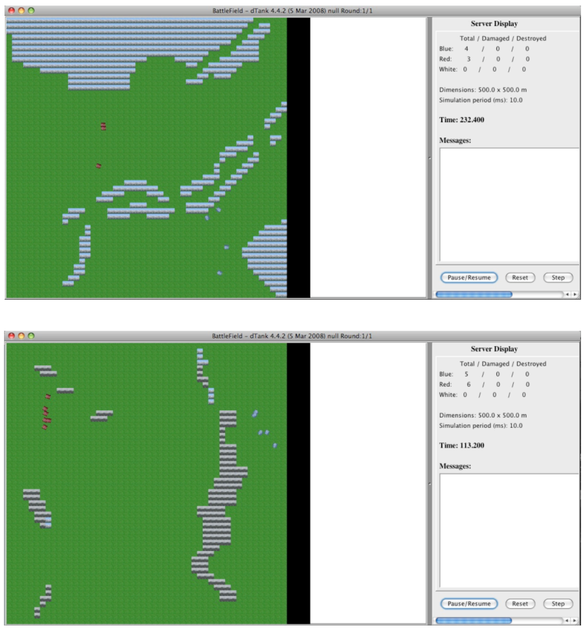


Figure 1. CoJACK in a Synthetic Environment.

5. Case Study of CoJACK's Variability

To explore these agent behaviours, we have been exercising them in dTank, a simple tank game that can be used as a lightweight synthetic environment (Ritter et al., 2007). Designed to compare agent architectures (Sun, Councill, Fan, Ritter & Yen, 2004), dTank has been used to examine how situation awareness might play out in a synthetic environment. Ritter et al. (2007) showed that situation awareness matters. Their report used two types of tank, a basic opponent tank and a tank with adjustable response delay. Their report showed that as response delay increased (in this case, from 0 to 3s), performance decreased and that the time range of expected effects of situation awareness on performance is supported by the simulation.

Here, we have created three 20 x 20 km maps for general comparisons using Sherman tanks. We are also working on an ambush scenario with a different map. The first terrain is a simple plain board. It functions as a base case and is useful in troubleshooting and debugging. The second and third simulate tank battles between German and British forces in World War II The battle of Medenine took place on a primarily open plain surrounded by hills (Figure 2, upper image). We chose this battle because an analysis of the details of how the battle was fought is available (Poncelin de Raucourt, 1997). The third terrain (Figure 2, lower image) is based on the battle of El-Alamein. This terrain contains primarily open ground with a few low hills. It is interesting because it is also a famous tank battle and provides a more complex domain due to the centre hills.



Total / Damaged / Destroyed
Blue: 4 / 0 / 0
Red: 3 / 0 / 0
White: 0 / 0 / 0
Dimensions: 500.0 x 500.0 m
Simulation period (ms): 10.0
Time: 232.400

Total / Damaged / Destroyed
Blue: 5 / 0 / 0
Red: 6 / 0 / 0
White: 0 / 0 / 0
Dimensions: 500.0 x 500.0 m
Simulation period (ms): 10.0
Time: 113.200

Figure 2. The maps used in the analyses.

To provide a fair comparison, the basic dTank tactics for each commander were taken from the simple default knowledge used to create previous tanks (Ritter et al., 2007). The basic subtasks are shown in Figure 3. This knowledge makes the commander capable of moving forward, turning, turret rotation, targeting and firing. This strategy relies on directed target search. After it has identified a target, the tank begins attacking. There remains some discussion about how uniformly this knowledge has been implemented in the tank commanders we present here, which we will have to take up after further analyses. When comparing architectures, the knowledge equivalence across models is important.

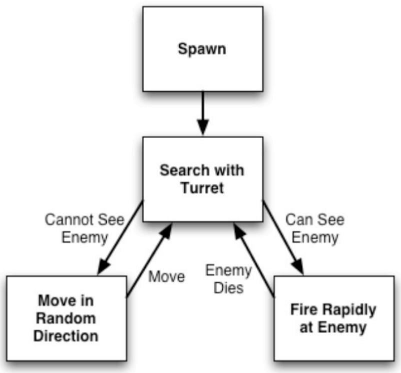


Figure 3. The default knowledge in the tank commanders.

We have only recently connected all these components, and are starting to examine the wealth of data dTank generates. The dTank system provides measures of: time to end of scenario, xy locations, number of shots and actions, variance in shots and actions, time between actions, number of kills and variance across runs and across tanks. From these measures, we can compute % of tanks destroyed, and efficiency of shooting. We also can examine change in performance across time for the more sophisticated tank commanders.

To start to illustrate the performance of CoJACK, we first ran Smart Java commanders against each other. These results are shown in Table 1a. They show that the tanks were fairly successful at shooting, and that the Java tanks were more predictable (smaller SD) than the CoJACK tanks (Table 1b). Figure 4 shows how the two of the tank commanders move (Java and CoJACK), based on an idea given to us by Sue Kase. The CoJACK tanks show less variance in their movement path and more intelligence. The Java commander has a more random path (and was killed sooner).

We are continuing to explore how these tanks performed, and what measures are of greatest interest. As we automate this drawing process we believe that we will see that the Java commanders have more variance in movement, and less variance in performance, and that CoJACK commanders have more variance in movement than the JACK commanders.

Table 1. Results for teams of 4 v 4 on empty 50 x 50 tile map on 1 Km^2 map, 10 runs. Standard deviations in parentheses. Other results include neither or both teams destroyed.

(a) Smart Java vs. Smart Java (4 v 4)

Team	Wins	Other	Tanks destroyed	Successful Shots	Shots
Red	3	2	32 (0.92)	138 (1.81)	156 (1.43)
Blue	5	2	34 (1.07)	147 (2.58)	150 (2.67)

(b) CoJACK vs. CoJACK.(4 v 4)

Team	Wins	Other	Tanks destroyed	Successfu l Shots	Shots
Red	2	0	19 (1.35)	110 (6.88)	241 (6.74)
Blue	8	0	34 (1.45)	187 (6.43)	253 (3.06)

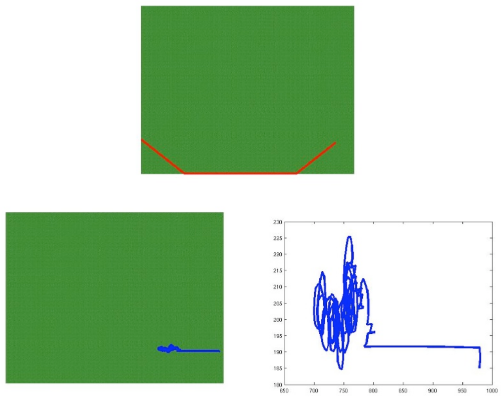


Figure 4. Tank movement patterns, CoJACK (top), and Smart Java (bottom, detail expanded on bottom right).

6. Discussion

CoJACK appears to be useful. It generates variable performance in a synthetic environment tied to theories of cognition and implemented with the usability of an agent architecture. It shows that theoretically grounded moderated behaviour in an agent architecture is possible, and that the generated behaviour can be interesting and productive. It, like other cognitive architectures, will need to be used more widely and more broadly before we fully understand it, but it is a promising architecture. It offers another platform for implementing the results of Silverman's review of moderators (Silverman, 2004).

We have also learned some things about variation. Variation in cognition does not necessarily lead to variation in behaviour, and we have seen that variation from the environment may lead to variation in a fixed agent, increasing the apparent range of performance. We have seen the need to include equivalent amounts of knowledge for fair comparisons, and also the need for running dTank in batch mode to generate summaries of behaviour.

At this point in time, we have the agent architecture (CoJACK), the agents (the knowledge noted above and the Java implementations for comparison), the environment (dTank), and some of the measures of interest. We are developing further measures of interest, further agents, better support from the environment, and better agent architectures. We will be exploring the moderators already implemented but not tested. We are now ready to study the effects of variability directly.

7. Acknowledgements

This work was supported by the UK MOD's Analysis, Experimentation and Simulation corporate research programme (Project No: RT/COM/3/006). We thank Colin Sheppard for useful discussions about CoJACK. Bil Lewis, along with Damodar Bhandikar, and Jeremiah Hiam, have helped develop this version of dTank. Comments and computations from Jon Morgan helped improve this presentation.

8. References

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?*. New York, NY: Oxford University Press.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Bach, J. (2007). *Principles of synthetic intelligence: Building blocks for an architecture of motivated cogntion*. New York, NY: Oxford University Press.

Bartl, C., & Dörner, D. (1998). PSI: A theory of the integration of cognition, emotion and motivation. F. E. Ritter, & R. M. Young (Eds.), *Proceedings of the 2nd European Conference on Cognitive Modelling,* 66-73.

Boff, K. R., & Lincoln, J. E. (1988). *Engineering data compendium: Human perception and performance*. OH: AAMRL, Wright-Patterson AFB.

Booher, H. R., & Minninger, J. (2003). Human systems integration in army systems acquisition. In H. R. Booher (Ed.), *Handbook of human systems integration*. (pp.663-98). Hoboken, NJ: John Wiley.

Bratman, M. E. (1987). *Intention, plans, and practical reasoning*. Cambridge, MA (USA): Harvard University Press.

Busetta, P., Rönnquist, R., Hodgson, A., & Lucas, A. (1999). JACK intelligent agents - components for intelligent agents in JAVA. *Agentlink News Letter, 2*(Jan.), www.agent-software.com/white-paper.pdf.

Georgeff, M. P., & Ingrand, F. F. (1989). Monitoring and control of spacecraft systems using procedural reasoning. *Proceedings of the Space Operations Automation and Robotics Workshop,*

Gratch, J., & Marsella, S. (2004). A domain-independent framework for modeling emotion. *Journal of Cognitive Systems Research, 5*(4), 269-306.

Howes, A., & Young, R. M. (1997). The role of cognitive architecture in modeling the user: Soar's learning mechanism. *Human-Computer Interaction, 12*, 311-343.

Hudlicka, E., & Pfautz, J. (2002). Proceedings of the eleventh conference on computer generated forces and behavioral representation. *Architecture and representation requirements for modeling effects of behavior moderators,* 9-20. 02-CGF-085.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

Kirlik, A. (2006). Abstracting situated action: Implications for cognitive modeling and interface design. A. Kirlik (Ed.), *Adaptive perspectives on human-technology interaction,* 212-224.

Lebiere, C., Biefeld, E., Archer, R., Archer, S., Allender, L., & Kelley, T. D. (2002). Proceedings of the advanced technologies simulation conference. *IMPRINT/ACT-R: Integration of a task network modeling architecture with a cognitive architecture and its application to human error modeling,*

Norling, E., & Ritter, F. E. (2004). A parameter set to support psychologically plausible variability in agent-based human modelling. *The Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS04),* 758-765.

Pew, R. W., & Mavor, A. S. (1998). *Modeling human and organizational behavior: Application to military simulations*. Washington, DC: National Academy Press. books.nap.edu/catalog/6173.html.

Pew, R. W., & Mavor, A. S. (2007). *Human-System integration in the system development process: A* *new look*. (R. W. Pew & A. S. Mavor, Eds.). Washington, DC: National Academy Press.

Poncelin de Raucourt, V. P. M. (1997). The reconstruction of part of the battle of medenine. *Unpublished MSc thesis*. Shrivenham, The Royal Military College of Science.

Silverman, B. G. (2004). Human performance simulation. In J. W. Ness, D. R. Ritzer, & V. Tepe (Eds.), *The science and simulation of human performance.* (pp.469-98). Amsterdam: Elsevier.

Sun, S., Councill, I. G., Fan, X., Ritter, F. E., & Yen, J. (2004). Proceedings of the sixth international conference on cognitive modeling. *Comparing teamwork modeling in an empirical approach,* 388-389.

Author Biographies

RICK EVERTSZ is a cognitive scientist at AOS Group. He is currently working on the incorporation of cognitive and affective constraints into BDI agents.

FRANK RITTER is on the faculty of the College of IST, an interdisciplinary academic unit at Penn State to study how people process information using technology. He edits the *Oxford Series on Cognitive Models and Architectures* and is an editorial board member of *Human Factors* and *AISBQ*.

PAOLO BUSETTA is a senior software architect at AOS Group. He has worked on numerous agent-based systems for research as well as military and commercial applications.

MATTEO PEDROTTI is a software architect at AOS Group, working on GUI design, and tracing and monitoring tools for agent architectures.

JENNIFER BITTNER is a graduate student in Cognitive Psychology and a research assistant in the College of IST at Penn State. She has a BA in Mathematics with a minor in Psychology from Penn State. Her current research examines face perception.