Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

ACTION MODELS

Gary W. King
Clayton T. Morrison
Paul R. Cohen

University of Massachusetts
Amherst, MA 01003, U.S.A.

ABSTRACT

Building military simulations requires bridging the gap between the knowledge of commanders and computer representations of that knowledge. A significant part of this knowledge concerns military tasks, their interactions, and an understanding of how to grade their achievement. *Action Models* describe the complex spatial and temporal dynamics of goal directed tasks with a graphical notation. Commanders can understand the notation and Knowledge Engineers can convert it into declarative or procedural forms. The conversion makes possible automated After Action reviews of plans writen in terms of these tasks (Center for Army Lessons Learned (CALL) 1993). We describe Action Models, their conversion into Tapir, a declarative executable action language, and their use in the DARPA Rapid Knowledge Formation (RKF) Program.

1 INTRODUCTION

Evaluating an action requires measuring and grading the performance of its sub-tasks. An Action Model describes the sequence of an action's expected events and goals using a simple graphical notation. Action Models idealize both the temporal and spatial aspects of an action. In doing so, they provide a bridge between the language of Subject Matter Experts (SMEs) and Knowledge Engineers.

Action Models consist of an abstract *graphical depiction* of an action's spatial interactions, *anchor points* (indicating transitions between phases of the action) and *attribute timelines*, specifying measurable attributes. For example, Figure 1 depicts the Action Model for the defeat task from Army Field Manual FM3-90 (Department of the Army 2001).

The Action Model for defeat contains three anchor points, dividing the defeat action into two phases: a maneuver phase and an attack phase. The anchor points define the beginning of the defeat action (AP_1), the transition from maneuver to attack when the enemy unit is encountered (AP_2), and the completion of the action (AP_3) when

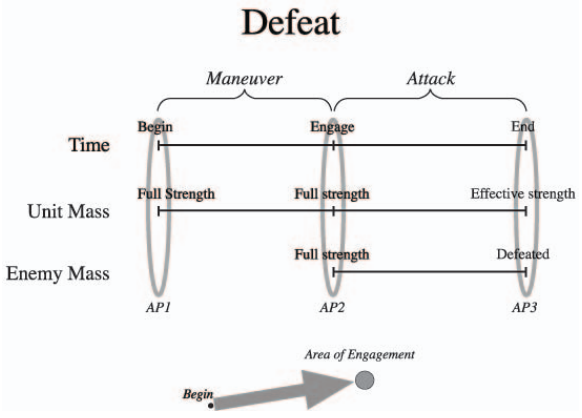

Figure 1: Action Model for the *Defeat* Task

the enemy is defeated, retreats, or the attacking unit is destroyed. Measurable attributes are time, the friendly unit(s) firepower, and the enemy's firepower. Time and friendly firepower can be measured at all three anchor points, but enemy firepower is only measured at the beginning of engagement and at the completion of the defeat action (AP_2 and AP_3, respectively).

Although Action Models are general and can be used to model any activity with well defined measurement criteria, we have concentrated our work in the military domain. In particular, our Action Model examples are drawn from Army Field Manual FM3-90 and have been used in Course of Action (COA) construction and evaluation.

This paper provides a description of Action Models and their use in knowledge engineering. Section 3 and 5 describe Action Models in detail, present several examples drawn from FM3-90, and indicate how Action Models can be converted into complete action specifications. We conclude with an overview of the DARPA Rapid Knowledge Formation (RKF) program and the role Action Models played in it (section 6). First, however, we compare Action Models with existing representations and highlight what makes them unique.

2 RELATED WORK

Though other action description languages exist, Action Models are unique in offering a graphical depiction of actions in terms of their phases and measurable goals. Languages such as ESL (Gat 1997) and TDL (Simmons and Apfelbaum 1998) implement actions whereas Action Models describe them. Other descriptive languages like DAML-S (Burstein et al. 2001) and PDDL (McDermott et al. 1998, McDermott 2000) are of little use to Subject Matter Experts because of their formality and abstraction. We have found that Action Models provide a bridge from natural SME descriptions to formal languages that can be computerized.

3 ACTION MODELS

Evaluating an action requires grading the performance of its sub-tasks. An Action Model describes the sequence of these tasks and the desired values of key attributes at their beginning and end. It also provides a spatial model of the task which represents abstractly how the tasks and the attributes are related.

Attribute measurements are graded with the five point scale (Figure 2) used in After Action Reviews (Center for Army Lessons Learned (CALL) 1993).

Figure 2: 5-point Grading Scale

The Action Model for defeat (Figure 1) shows the desired attribute values at each anchor point. For example, the unit's firepower should be at least *effective* and the enemy's firepower should be *defeated* at the end of the action. Different attributes require different kinds of grading scales. Two useful scales in the military domain are mass and timing. Mass is an abstraction of all the characteristics of a force that make it effective. It includes force size, weapon type, cohesion, morale, training, experience and so on. Timing is the coordination of tasks relative to a single action and to a plan as a whole.

3.1 Mass Grading Scales

Mass scales measure the effectiveness of military units relative to their initial strength. The effectiveness of a unit depends both on its percentage of remaining mass and on how well it can cope with the loss of that mass. For example, elite units will continue to function effectively with much higher casualty levels than green troops. To account for this, we use separate scales for unmotivated, veteran and elite troops. Figure 3 depicts the scale for veteran troops.

Each scale maps the percentage of remaining mass to a military effectiveness level: full strength, neutralized, defeated and so on.

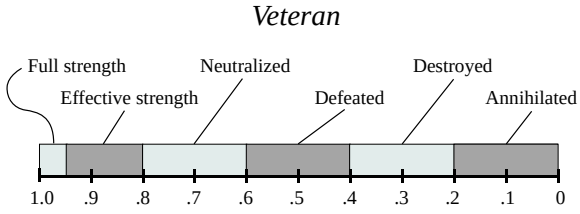

Figure 3: Mass Scale for Veteran Forces

The labels on the mass attribute timelines (Figure 1) specify desired effectiveness. A unit at this level at would receive a grade of `OK` (Figure 2) – if the unit is at another level, its grade would depend on whether it is a friendly or enemy force. For example, if a friendly blue unit is trying to defeat an unfriendly red one; then the final goal (AP_3 in Figure 1) is to have the blue unit at `effective strength` and the red unit at `defeated`. For veterans, the 5-point scale is calibrated so that a final mass between 80 and 95% of initial mass will receive a score of `OK`. Mass ranges around this base take on other scores. For example, because we are measuring friendly mass, levels higher than 95% will receive a score of `+`, but if mass is in the `neutralized` range (60 to 80%), a score of `-` will be assigned. Even lower masses will receive a `–`. On the other hand, if we are measuring desired negative effects on an enemy unit, and the goal is to reduce its mass to `defeated`, then an `OK` would be assigned if the final enemy unit mass is between 40 and 60%, a `-` if between 60 and 80% (neutralized), a `–` if between 80 and 95% (effective strength), `+` if between 20 and 40% (destroyed), and `++` if between 0 and 20%.

3.2 Time Grading Scales

Inter- and intra-action coordination is assessed using time scales. Time may be measured on an absolute scale or relative to the timing of sub-tasks. As mentioned, Action Models have internal structure based on transitions between sub-tasks. For example, the defeat action has a beginning, an engage-event, and an end – these are the events which a commander can use to coordinate with other events or actions in a COA. A *time grading scale* is a way of asserting and assessing coordination goals.

The scale provides an envelope of time intervals around some specific time t. Grades are assigned to each interval according to the timing relationship the envelope represents. With these grades, the envelope describes the commander's desire for how close to t some event *should* occur, keeping in

mind that there are many variables that may affect whether timing goals specified in a COA can be met in actual practice. Time *t* is a general temporal marker: it may refer to absolute time (e.g., related to specific hour times), to coordination times defined in a COA specification (e.g., by *H*-hour), or to other events (e.g., the completion of action-*b*, the arrival of unit-2, or the crossing of phase-line *Lima*, etc.).

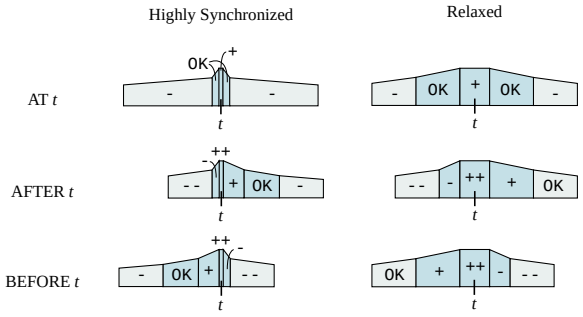

Figure 4: Time Grading Scales

Figure 4 depicts grading scales for three types of timing relationships between *t* and an event, each with two settings. A seventh option is to assert that timing does not matter for the event. The scales in Figure 4 indicate that the event should happen at *t*. The score received depends on the tightness of the scale, how distant the event is from *t*, and whether the event occurs before or after *t*. If it occurs at *t*, it will be scored as `OK`.

3.3 Abstract Spatial Representations

Action Models also include an depiction of the spatial relationships which the action ought to maintain or which can be used to understand the relationship between subtasks. The defeat action has a very simple model showing that the friendly unit must first travel to the enemy and then engage it. Other models have more complex depictions.

3.4 Action Model Examples

The next several sub-sections provide examples of Action Models. Each example begins with a short italicized paragraph describing the intent of the action. These are paraphrased from Appendix B of Field Manual FM3-90.

3.4.1 Block

Deny enemy access to an area or prevent advance in a direction or along an avenue of approach.

A block action is defined with respect to a *Block Line* and a direction (Figure 5). The goal of the block is to prevent any enemy forces advancing from the direction from crossing the *Block Line*.

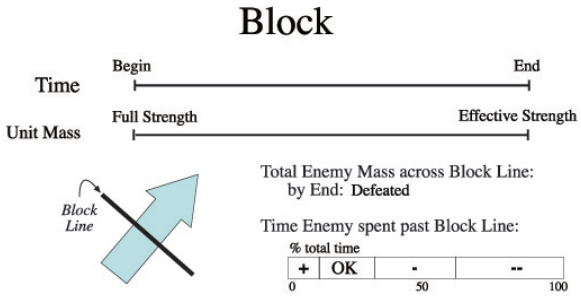

Figure 5: The Block Action Model

Beginning and end times can be measured, as well as unit mass (which has a desired end-mass of `effective strength`). During the block, any enemy units that cross the *Block Line* are tracked. If they are pushed back across the line in the direction of the block, then they are considered successfully blocked. Two attributes measure the success of the block. At the end of the action the total enemy mass across the block line should be `defeated` or below. The intuition here is that even if an enemy unit did manage to get across the line, the block is still successful if it is no longer effective. The other attribute tracked is the percentage of time enemy forces spent across the *Block Line* relative to the duration of the block. The scale in Figure 5 specifies the grading for this time. This is measured by tracking the time that any enemy units are across the line.

3.4.2 Fix

Prevent enemy from moving any part of his force from a specific location for a specific period.

In a fix action (Figure 6), the commander specifies the enemy unit(s) to be fixed (as the spatial model indicates, if more than one unit is selected, they must be close enough to fix simultaneously).

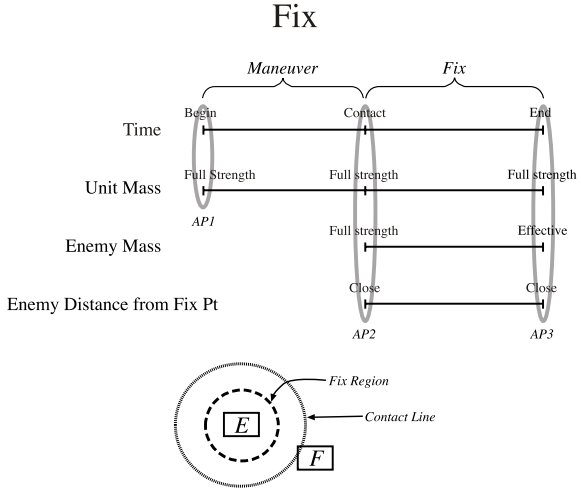

Figure 6: The Fix Action Model

A fix involves two phases – maneuvering to the location of the enemy, and holding the enemy in place. Two spatial regions are relevant to fix. The *fix region* is the area in which the enemy should be held. The *contact line* is the outer-most distance from the target for which the fixing unit can still affect the target unit.

A fix action is not intended to strongly engage its target; rather the goal is to keep the target from moving from its position. This is accomplished by threatening attack and engaging if the target moves, exposing a vulnerable flank or rear. For this reason, the target may still be at *effective strength* at the end of the action. The key criterion being measured is the distance from the fix point. *Close* is determined by the fix region, and grading is with respect to relative distance from this region that the target manages to reach; if the target remains in the region, the fix receives an `OK`; if it leaves the region, a - or – is assessed depending on how far the unit has traveled.

3.4.3 Screen

A unit performing a screen *observes, identifies, and reports enemy actions. Generally, a screening force engages and destroys enemy reconnaissance elements within its capabilities – augmented by indirect fires – but otherwise fights only in self-defense. The* screen *has the minimum combat power necessary to provide the desired early warning, which allows the commander to retain the bulk of his combat power for commitment at the decisive place and time. A* screen *provides the least amount of protection of any security mission; it does not have the combat power to develop the situation. The* screen *should allow no enemy ground element to pass through the screen undetected and unreported.*

As depicted in Figure 7, a screening force is associated with some larger force some distance behind it, so a screen action must coordinate with the larger force.

The screening force is small, and its primary mission is intelligence gathering and counter-intelligence operations. It will only engage small forces being used for intelligence purposes by an opponent – for our purposes, we can assume that it will only attack small forces for which it has a better than 1-1 combat power ratio. The large arrow in the figure depicts the direction *from* the force that the screening unit is in service of. Throughout the screen, the screening unit may move within the screening perimeter as necessary in order too maintain contact with any enemy units, while also keeping a safe distance to avoid engaging a larger enemy force.

The two main attributes measured are the time for which the screen is to be in place (absolute or relative to other events (e.g., completion of preparation of the force the screen is in service of), and the mass of the screening unit). The screening action has two phases, a travel phase

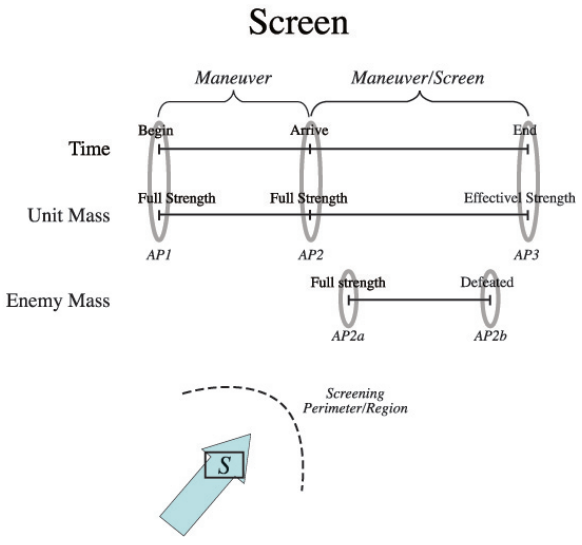

Figure 7: The Screen Action Model

which ends when the unit arrives at the screening location (AP_2), and maneuvering to patrol the area. The screening unit should still be at effective strength by the end of the screen. The enemy mass attribute scores how effectively the screening unit managed counter-intelligence operations, if any.

3.4.4 Breach (Penetrate & Exploit)

Use all available means to break through or secure a passage through an enemy defense, obstacle, minefield or fortification.

A breach (Figure 8) is an action model that involves explicit coordination between two units: a *penetrating* force and an *exploiting* force.

The penetrating force is tasked with engaging an enemy force in order to create an opening through which the exploiting force may safely pass. The diagram at the bottom sketches what this looks like. The penetrating force (indicated by *P*) overlaps the target enemy unit(s) and creates an opening. Meanwhile, the exploiting force (*E*) maneuvers into position and waits. When the enemy is breached, it moves through the opening. The direction of the movement, indicated by the arrow, is assumed to be in the direction opposite where the penetrating and exploiting force began their action.

A breach action has three phases for each unit (for four anchor points each). The penetrating unit maneuvers to the enemy, engages it to create an opening, and continues to engage to maintain the opening. The exploiting unit maneuvers into a waiting position, waits and then maneuvers through the opening once it is created. All of the anchor points for either friendly force are synchronized except for AP_{2-a} and AP_{2-b} which may happen at different times (in the Figure AP_2 is reach after AP_{2-a}, but this is not

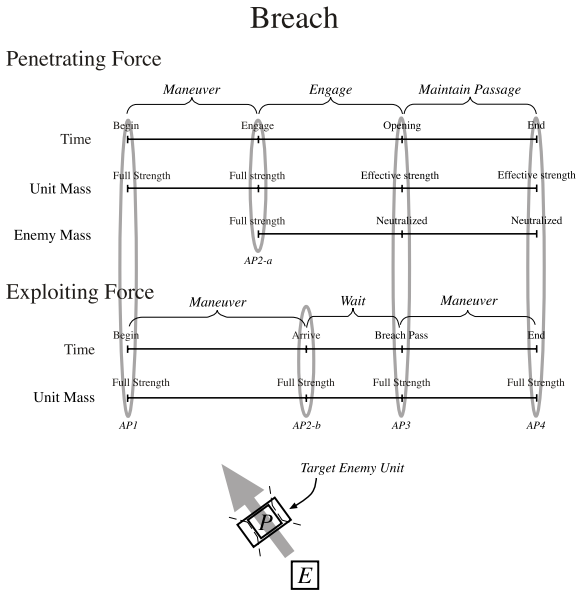

Figure 8: The Breach (Penetrate & Exploit) Action Model

necessary); the only constraint is that the exploiting force achieve the best position for exploitation before the penetrating force creates the opening. While the opening is present, it is assumed the target enemy will not be able to affect the exploiting unit, although it may still affect the penetrating force. The goal of the breach is the exploitation of the opening by the exploiting force, not the destruction of the enemy, so the goal is to just neutralize the enemy. The exploiting force should also be at full strength by the end of the breach action.

4 TAPIR

Of course, Action Models describe only the normative goals and temporal phasing of a task. They do not describe resources, paths, manner or contingency. They are the first step of formalizing tasks and provide a common framework in which to do so. We have used Tapir (King et al. 2002, King et al. 2002) to construct executable implementations of Action Models from FM3-90.

Tapir is a general purpose activity language for describing behaviors and goal-directed tasks. Tapir's ontology is built around four hierarchies: action, sensor, goal and resource.

Actions: Actions use resources and sensors and may satisfy goals. Each action exists in the context of its parent action and may extend the hierarchy by creating children of its own. The action hierarchy is grounded in primitive actions that use resources to achieve their ends. Multiple actions run concurrently as separate processes.

Sensors Sensors tie actions to the world. They can be primitive, connecting to the world directly, or abstract, amalgamating and processing data from multiple sources. Sensors can be shared by multiple actions.

Goals: Goals describe connections between predicates about the world and actions that can satisfy the predicates. Goals also specify satisfaction criteria that score how well they have been achieved. Tapir actions describe their costs and how to prioritize them which helps planners select an optimal plan for a goal set. Tapir includes a reactive planner to match goals with actions that satisfy them.

Resources: Resources connect actions to the physics of the simulation in order to alter it. Note that we use the term "resources" for the things that let an action do its work. Thus Tapir does not view gasoline as a resource for a car. Instead, the gasoline level is a property of the car and the engine is a resource for the car's movement action. Resources can be the effectors of a single agent or the agents themselves. To Tapir, an agent is a coherent group of effectors. For example, a robotic agent is a resource consisting of motor effectors, camera effectors and a gripper effector. Any subset of these effectors can be bound for use by an action. Depending on the domain, resources can be serializably reusable, sharable, composable, consumable and so on.

Tapir is a message-based language. Communication between actions, sensors, goals and resources occurs only via messages. For example:

- sensors send messages to actions when events in the world occur.
- child actions send messages to their parents when they complete or when they need to communicate status changes.
- parents send their children messages when they wish to stop or redirect their activities.
- resources send messages to their actions when they change or are destroyed.

Each message is an instance of a particular class and can carry additional information. For example, a failure message carries the reason for its failure and a change-speed message carries the value of the desired speed.

5 FROM ACTION MODELS TO ACTIONS

Action Models represent actions as ordered subtasks and attribute goals measured at the transition between subtasks. Tapir represents actions as sets of interacting subtasks and represents goals as predicates about the world along with scoring criteria. Tapir-represented actions can be executed in simulation and evaluated using Monte Carlo techniques.

```
(defaction defeat ()
  (:documentation "Move to and engage an enemy
                           in order to defeat them.")
  (:resources ((agent :type military-agent)))
  (:parameters (target :type military-agent))
  (:do
    (:achieve (move-within-attack-range-goal
               :target target))
    (:achieve (reduce-enemy-mass-goal
               :goal-target-mass +mass-defeated+
               :target target)))
  :send-messages-to-parent)

;;; categorical mass is a measure which converts an
;;; agent's absolute mass (a number) into a
;;; descriptive military level.
(defmeasure categorical-mass ()
  (:parameters (agent))
  (:value (if agent
              (categorical-mass-lookup
                  (experience-level agent) agent
                                        (mass agent))
              +mass-annihilated+))
  (:ordinal +mass-full-strength+ +mass-effective+
            +mass-neutralized+ +mass-defeated+
            +mass-destroyed+ +mass-annihilated+)
  (:direction :decreasing))

(defgoal move-within-attack-range-goal ()
  (:resources ((agent :type military-agent)))
  (:parameters ((target :type spatial-region :required)))
  (:satisfied-p (in-attack-range-p agent target)))

(defgoal reduce-enemy-mass-goal ()
  (:resources ((agent :count :all)))
  (:parameters ((goal-target-mass +mass-destroyed+)
                (target :required)
                (goal-engagement-mass
                           +mass-full-strength+)))

  (:score :enemy-mass
          (cond ((= (categorical-mass target)
                    goal-target-mass)
                 +ok+)
                ((comes->= (next (categorical-mass target)
 2) goal-target-mass)
                 +double-plus+)
                ((= (next (categorical-mass target))
                    goal-target-mass)
                 +single-plus+)
                ((= (previous (categorical-mass target))
                    goal-target-mass)
                 +single-minus+)
                ((comes-<= (previous (categorical-mass
                            target) 2) goal-target-mass)
                 +double-minus+)))

  (:score :friendly-mass
          ... <similar to enemy-mass score ...)

  ;; The goal is satisfied when the goal-target-mass is
  ;; greater than the current categorical mass of the
  ;; target. For example, if the goal is to have the
  ;; enemy defeated, then the goal is satisfied when the
  ;; enemy's level is defeated or lower.
  (:satisfied-p (comes->= goal-target-mass
                           (categorical-mass target))))
```

Figure 9: The defeat Task Written in Tapir

Figure 9 presents an implementation of the defeat Action Model in Tapir. This consists of one action, two goals and a function to convert numeric mass into categorical mass labels. The *defaction* statement defines defeat as achieving two goals in order: moving within range and then reducing enemy mass. The two *defgoal* statement define these goals. Note that the plans which achieve these goals are not presented. These are other actions defined by *defaction*. The reduce-enemy-mass-goal provides two scoring functions: one for enemy mass and one for friendly mass. Note that this goal can be used by multiple actions and its score will depend on the actual target mass passed in as a parameter to the goal. The bulk of the scoring function is concerned with determining where the final categorical enemy mass lies in comparison to the goal mass in order to provide a final score somewhere between ++ and –.

6 CASE STUDY

The DARPA Rapid Knowledge Formation (RKF) Program's central goal is to enable distributed teams of subject matter experts (SMEs) to enter and modify knowledge directly and easily, without the need for specialized training in knowledge representation, acquisition, or manipulation (Clark et al. 2001, Barker et al. 2003). The Summer 2002 evaluation of RKF focused on military Course of Action analysis.

A COA is a plan outline used by a commander to coordinate and communicate with his or her subordinates. A complete description of the COA critiquing criteria can be found at http://www.iet.com/Projects/RKF/COA-Critiquing-Criteria.ppt. Several different COAs are considered for each mission; they are evaluated along a variety of situation-dependent dimensions such as complexity, risk, speed, logistical support and so forth. Part of this evaluation typically consists of subjective "war gaming" of each COA by the commander's staff. The RKF challenge was to formalize COA critiquing by producing critiques based on a graphical depiction of the COA produced with Northwestern University's nuSketch Battlespace application (Rasch et al. 2002, Forbus et al. 2003) and a narrative produced by an SME. The teams within RKF used natural language input as part of the Cyc project (Lenat 1995), newly developed tools such as the Concept Map (Ca-as et al. 1999) inspired SHAKEN project (Uribe et al. 2002), tools for normative simulation such as KANAL (Kim and Gil 2001) and empirical simulation tools such as Capture the Flag(CtF) (Atkin et al. 2000). Normative simulation executes a COA one step at a time as if each task accomplishes what it is "supposed to". It provides a rapid critique of specification errors such as failed preconditions and missing steps. Empirical simulation, on the other hand, is computationally more expensive but it both discovers emergent interactions – the things you did not expect – and more easily handles the time based interactions of multiple concurrent processes.

Action Models were used to communicate between SMEs and the KEs and simulator builders at the University of Massachusetts and, to a lesser extent, with other members the RKF effort. Though no formal measures exist, we found them to be an efficient means of communication and formalization. In particular, converting Action Models into

Tapir is a fairly routine process. Of course, more complex action models with greater degrees of contingency were correspondingly more difficult to Tapirize.

7 SUMMARY AND FUTURE WORK

We have presented Action Models, a novel formulation of activity. Action Models describe actions with a spatial model, a set of anchor points (dividing sub-tasks) and a set of measurable attribute timelines (describing goals). We have shown how Action Models can be readily formalized in an action description language such as Tapir. Our current work involves building tools to acquire Action Models from directly from SMEs and tools to combine them together into complete, executable COAs (King and Hannon 2003).

ACKNOWLEDGMENTS

We thank General Charley Otstott, for inspiring this work and for the careful and insightful criticism he brings to the task. This research is supported by DARPA/USAF under contract numbers F30602-01-1-0589, N66001-00-C-801/34-000TBD, and F30602-99-C-0061, The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the Defense Advanced Research Projects Agency/Air Force Materiel Command or the U.S. Government.

REFERENCES

Atkin, M. S., et al. 2000. SPT: Hierarchical agent control: A framework for defining agent behavior. In *Proceedings of the Fifth International Conference on Autonomous Agents*: Autonomous Agents.

Barker, K., et al. 2003. A knowledge acquisition tool for course of action analysis. In *Innovative Applications of Artificial Intelligence*.

Burstein, M., et al. 2001, May. DAML-S 0.5 draft release (available at http://www.daml.org/services/daml-s/2001/05/). Technical report, DARPA.

Ca-as, A., D. Leake, and D. Wilson. 1999. Managing, mapping, and manipulating conceptual knowledge. Technical report, The University of West Florida.

Center for Army Lessons Learned (CALL) 1993. A leader's guide to after-action reviews. Technical report, Fort Leavenworth, KS.

Clark, P., et al. 2001. Knowledge entry as the graphical assembly of components. In *Proceedings of the international conference on Knowledge capture*, 22–29: ACM Press.

Department of the Army 2001. Field manual 3-90: Tactics. Technical report.

Forbus, K. D., J. Usher, and V. Chapman. 2003. Sketching for military courses of action diagrams. In *International Conference on Intelligent User Interfaces*.

Gat, E. 1997. ESL: a language for supporting robust plan execution in embedded autonomous agents. In *Proceedings of the IEEE Aerospace Conference*, 319–324. Snowmass at Aspen, CO, USA.

Kim, J., and Y. Gil. 2001. Knowledge analysis on process models. In *IJCAI*, 935–942.

King, G., and A. Hannon. 2003. The plan Acquisition Tool: A tutorial. Technical report, University of Massachusetts Computer Science.

King, G. W., M. S. Atkin, and D. L. Westbrook. 2002. Tapir: the evolution of an agent control language. In *Computers and Games 2002*.

King, G. W., M. S. Atkin, D. L. Westbrook, and P. R. Cohen. 2002. Tapir: an action language beyond scripting. In *The 3rd International Conference on Computers and Games (CG'02)*.

Lenat, D. B. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38 (11): 33–38.

McDermott, D. 2000. The 1998 AI planning systems competition. *The AI Magazine* 21 (2): 35–55.

McDermott, D., et al. 1998. The PDDL planning domain definition language (available at http://www.cs.yale.edu/). Technical report, Yale Computer Science Department.

Rasch, R., A. Kott, and K. Forbus. 2002. AI on the battlefield: an experimental exploration. *Innovative Applications of Artificial Intelligence*.

Simmons, R., and D. Apfelbaum. 1998. A task description language for robot control. In *Proceedings of Conference on Intelligent Robotics and Systems*.

Uribe, T. E., V. Chaudhri, P. J. Hayes, and M. E. Stickel. 2002, mar. Qualitative spatial reasoning for question-answering: Axiom reuse and algebraic methods. In *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*. Stanford, CA: AAAI.

AUTHOR BIOGRAPHIES

GARY W. KING received his B.A. in Chemistry from Wheaton College in Illinois, his Masters in mathematics from the University of Pennsylvania and his Masters in Computer Science from the University of Massachusetts. Currently, he is a Senior Research Fellow at the University of Massachusetts, Amherst in the Experimental Knowledge Systems Laboratory under the direction of Paul R. Cohen. His research interests include programming language design, simulation, language acquisition and algebraic information theory. If he had spare time, he would spend it with his wonderful wife and children. His email

and web addresses are <gwking@cs.umass.edu and <www.cs.umass.edu/~gwking/>.

CLAYTON T. MORRISON is a postdoctoral research fellow in the Department of Computer Science at the University of Massachusetts and a member of the Experimental Knowledge Systems Laboratory, directed by Paul Cohen. He received his B.A. in Cognitive Science from Occidental College and his M.A. and Ph.D. in Philosophy from the State University of New York, at Binghamton. His research interests include the nature of representation and knowledge in humans and machines, cognitive development, animal and machine learning, analogical cognition, and simulation. He is currently working on the Robot Baby project, which models cognitive development in a mobile robot, and a Bayesian blackboard system for the interpretation and analysis of asynchronous and noisy data from a variety of complex problem domains. His email and web addresses are <clayton@cs.umass.edu> and <www.cs.umass.edu/~clayton/>.

PAUL R. COHEN received his Ph.D. from Stanford University in 1983. He is a Professor of Computer Science at the University of Massachusetts, and Director of the Experimental Knowledge Systems Laboratory. He edited the "Handbook of Artificial Intelligence," Volumes III and IV with Edward Feigenbaum and Avron Barr. Cohen was elected in 1993 as a Fellow of the American Association for Artificial Intelligence, and served as Councillor of that organization, 1991–1994. His research concerns the design principles for intelligent agents and the acquisition of conceptual structures. His email and web addresses are <cohen@cs.umass.edu> and <www.cs.umass.edu/~cohen/>.