A Knowledge Acquisition Tool for Course of Action Analysis*

Ken Barker[1], Jim Blythe[2], Gary Borchardt[3], Vinay K. Chaudhri[4], Peter E. Clark[5], Paul Cohen[6], Julie Fitzgerald[9], Ken Forbus[7], Yolanda Gil[2], Boris Katz[3], Jihie Kim[2], Gary King[6], Sunil Mishra[4], Clayton Morrison[6], Ken Murray[4], Charley Otstott[8], Bruce Porter[1], Robert C. Schrag[9], Tomás Uribe[4], Jeff Usher[7], Peter Z. Yeh[1]

1. University of Texas at Austin 2. Information Sciences Institute at University of Southern California 3. Massachusetts Institute of Technology 4. SRI International 5. The Boeing Company 6. University of Massachusetts at Amherst 7. Northwestern University 8. Retired Lieutenant General, U.S. Army 9. Information Extraction and Transport Corporation

Abstract

We present the novel application of a general-purpose knowledge-based system, SHAKEN, to the specific task of acquiring knowledge for military Course of Action (COA) analysis. We show how SHAKEN can capture and reuse expert knowledge for COA critiquing, which can then be used to produce high-level COA assessments through declarative inference and simulation. The system has been tested and evaluated by domain experts, and we report on the results. The generality of the approach makes it applicable to task analysis and knowledge capture in other domains. The primary objective of this work is to demonstrate the application of the knowledge acquisition technology to the task of COA analysis. Developing a system deployable in an operational environment is the subject of future work.

Introduction

The goal of the SHAKEN project is to let subject matter experts (SMEs), unassisted by AI technologists, assemble models of mechanisms and processes from components. Questions about these models can be answered both by conventional inference methods, such as theorem proving and taxonomic inference, and by more task-specific methods, such as simulation and analogical reasoning. We believe that the assembly of components instantiated to a domain is a natural way for SMEs to create knowledge base content.

This paper describes the application of SHAKEN to the acquisition and use of knowledge needed for military Course of Action (COA) analysis. We begin with a technical overview of SHAKEN. We then describe the COA application, and give an overview of its solution using SHAKEN. For each aspect of the solution, we describe the technical challenges faced, and how we addressed them. We conclude with an evaluation of our approach, and directions for future work.

Functional Design of SHAKEN

The SHAKEN system has the following functional units, shown in Figure 1: a knowledge base (KB), an interface for entering knowledge, a set of tools for verifying and using knowledge, and a Web-based interaction manager. The KB, also called the *component library*, or CLIB [3], is a collection of components representing (a) general knowledge about common physical objects and events, states of existence, and core theories, including time, space, and causality, and (b) more specialized knowledge about particular domains, including micro-biology, chemistry, military units, military equipment, and terrain.

By a component, we mean a coherent set of axioms that describe some abstract phenomenon (e.g., the concept of *invade*) and are packaged into a single representational unit. Our claim is that a small number of predefined components is sufficient to let SMEs assemble models of virtually any mechanism or process. These components are mostly domain independent, but their assembly and specialization can create domain-specific representations.

The main task of the knowledge entry interface is to let SMEs assemble the right KB components, by connecting predefined elements of the component library. This is performed through a graphical interface, where SMEs assemble components by manipulating graphs. Axioms are automatically derived from the graphical representation, so the SMEs do not have to be trained in formal logic [8].

SHAKEN supports several different methods for using knowledge. Declarative inference, performed using the Knowledge Machine knowledge representation system (KM) [7], is the most common approach for using knowledge. Normative simulation is used to exercise the process knowledge in the system [17]. It executes each step in the process and analyzes interdependencies. Empirical simulation exercises knowledge by running a detailed simulation of a process using the Capture the Flag simulation engine [1]. An analogical reasoner, based on the Structure-Mapping engine [9], computes similarities

* For more information about the SHAKEN system, contact Vinay Chaudhri at chaudhri@ai.sri.com

and differences given two concept representations [21]. These methods can be invoked by a variety of means included in the question-asking interface [6]. The answers to questions are returned in an easily understood format, and the user can control the level of detail in an answer.

The interaction manager is aimed at making the knowledge entry experience seem natural. It handles limited forms of natural language input, and keeps track of the history of a knowledge acquisition session. A knowledge analysis module and an analogy module support the interaction manager and let SHAKEN take the initiative in helping an SME enter knowledge [17]. For example, the knowledge analysis module helps users verify and validate their process descriptions by analyzing the results from normative simulation. The vision for the interaction manager is to make the knowledge entry similar to a student/teacher interaction, where both the user and the system take the initiative at different times [19].

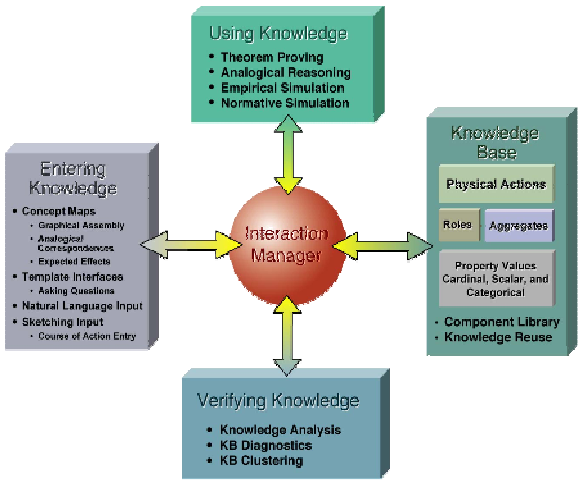


Figure 1: SHAKEN functional architecture

The KB server provides facilities for efficient storage and access of knowledge, based on KM [7]. It stores both domain-independent and domain-specific knowledge.

Knowledge verification based on normative simulation is used during knowledge entry by SMEs. KB clustering and diagnostics are used off-line both to support the development of domain-independent knowledge, and to do a post-hoc analysis of the knowledge entered by the SME.

Task: Course of Action Analysis

A military COA is a plan outline used by a commander to communicate to his subordinates one way to accomplish a mission. Normally, commanders consider several different ways to accomplish a mission, that is, several different COAs. They evaluate competing COAs using appropriate comparison criteria and decide on one to build into a complete action plan for the mission. In this paper, we consider COAs for ground military forces conducting offensive (attack) operations. The detail captured in the COA depends on the echelon. We consider here COAs at the level of a military division, a brigade, or a batallion. We consider only the COAs of friendly forces. Possible COAs for the enemy forces are not considered.

A COA specification is formulated in response to a specific situation between opposing forces and a mission directive. For purposes of description, we organize a COA specification into two parts: problem statement and solution statement. A COA problem statement consists of the following: (1) a situation sketch (on a map), indicating terrain features such as roads, rivers, lakes, hills, forests, and current Blue and Red unit placement; (2) a scenario narrative, including any details not easily captured on the map (e.g., relevant recent history, current dynamics, expected future evolution, unit status descriptions); (3) a mission specification, indicating specific forces under command, required objectives, and constraints (e.g., "Capture Objective JAYHAWK by 1400 hours tomorrow with the following restrictions in place…"); and (4) the commander's estimate of the situation.

Faced with such a problem statement, a commander must formulate a plan for his forces to accomplish the mission. He considers one or more options, or COAs. A COA solution consists of: (1) a COA sketch—an overlay on the problem statement's situation sketch, and (2) a COA narrative—a structured description stating the mission, commander's intent, desired end state, and the concept of operations, including main attack, supporting attack, fire support, and reserve. Each task in the COA must indicate what units perform what actions for what purposes.

Given enough time to consider alternatives, the commander's staff evaluates the candidate COAs in a subjective critiquing process, usually resulting in a matrix comparing the viable ones, and presents the results to the commander for a decision on the preferred COA. Commonly used COA-critiquing criteria include mission accomplishment, reserve availability, speed, simplicity, terrain use, risk, and position for follow-up operations. With help from domain experts, we created an extensive taxonomy of critiquing criteria. The COA critiquing task is to evaluate a formally represented COA with respect to key critiquing criteria. The purpose of critiquing and comparing different COAs is to help the commander decide how best to accomplish the assigned mission.

Given this definition of the COA analysis problem, the tasks to be performed were twofold: (1) given textual and graphical COA problem statements, formally represent selected elements of these in a knowledge base, and (2) author (conceive of and formally represent) knowledge to support effective COA critiques, which can then be applied to any formally represented COA solution statement.

We now briefly consider the possible deployment of a COA critiquing system. The critiquing knowledge will be entered in an Army laboratory long before the system is actually used in the field. The COA problem and solution statements will be entered at the time of actual usage of the system. Thus, when the critiquing task is performed in response to an actual need, the relevant critiquing knowledge will already be available. Given that we were developing an initial prototype, the task of entering COA problem and solution statements, and the task of authoring critiquing knowledge, are interleaved much more than they might in a situation when a COA critiquing system has been built and deployed.

Solution: Using SHAKEN to Acquire and Apply COA Critiquing Knowledge

As stated in the previous section, the overall task has two main aspects: COA authoring, and COA critiquing. With reference to the functional architecture of Figure 1, the tasks of authoring the COA and the critiquing knowledge are supported by the knowledge entry subsystem. COA authoring relies on battlespace knowledge that is built into the knowledge base. The SME enters the critiquing knowledge during development, which is stored in the knowledge base. The module focused on using knowledge supports the critiquing task. The interaction manager and the knowledge verification module play a supporting role in the overall solution of the problem.

COA Authoring

To formally author a COA, we needed to solve two problems: (1) provide a vocabulary of terms that can be used in COA authoring, and (2) provide a natural user interface for commanders.

Vocabulary for COA authoring: To support COA authoring, we need to represent military units, terrain, and military tasks. For military tasks, we developed two different representations: one suitable for declarative inference, and the other suitable for empirical simulation. Let us consider these two in more detail.

To develop representations for knowledge analysis, we leveraged the domain-independent representations in the component library to provide military-specific terms. For example, consider the military task *Canalize*. This is a tactical mission task where a military unit restricts enemy movement to a narrow zone. We represented this domain-specific action by specializing the domain-independent action *Confine*. The *Canalize* task differs from *Confine* in that its *agent* and *object* are military units, and its *base* is a piece of narrow terrain. It is similar to *Confine* in that its base plays the role of a container, and the object is inside the base after the action has been performed.

Empirical simulation requires a model of the domain and a model of the processes that occur in that domain. Our domain model is built on the University of Massachusetts Abstract Force Simulator (AFS) [2]. Military engagements are represented using circular agents moving on a coarse representation of real terrain. The agents have many properties, but most of the ones significant to military modeling (training, weapons type, troop strength, experience, and so on) are agglomerated into a single property: mass. The process model represents actions as lists of desired effects on key properties. Figure 2 shows the action model for *Defeat*, which is broken into two phases: one for the friendly forces to reach the enemy and one for the engagement. Each phase has corresponding goals for the action. The action models for the military tasks in the field manual are represented within AFS using Tapir, a general purpose, semi-declarative hierarchical agent control language that can express goals, sensors and actions using a unified syntax [18]. During each simulation run, the action models control the military agents; dynamically reacting to the changing properties of the simulation in order to achieve their goals.

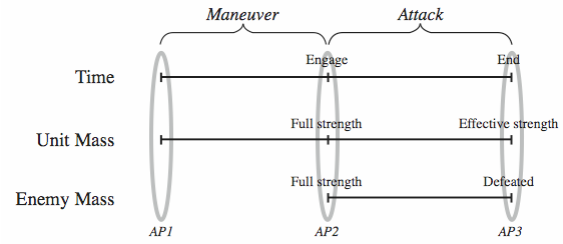


Figure 2: Action model for *Defeat*

User interface for COA authoring: We needed an interface that was as familiar to commanders as possible. Commanders work with maps and overlays to show the geography, unit locations, and military tasks. The map is usually accompanied by a textual description. The nuSketch system is explicitly designed to support COA authoring, and met this requirement very well [12], [13].

NuSketch provides a graphical interface where COA terrain, units, avenues of approach, and tasks can be described. The user can also specify the commander's intent for the overall COA and individual tasks. An example COA sketch is shown in Figure 3.

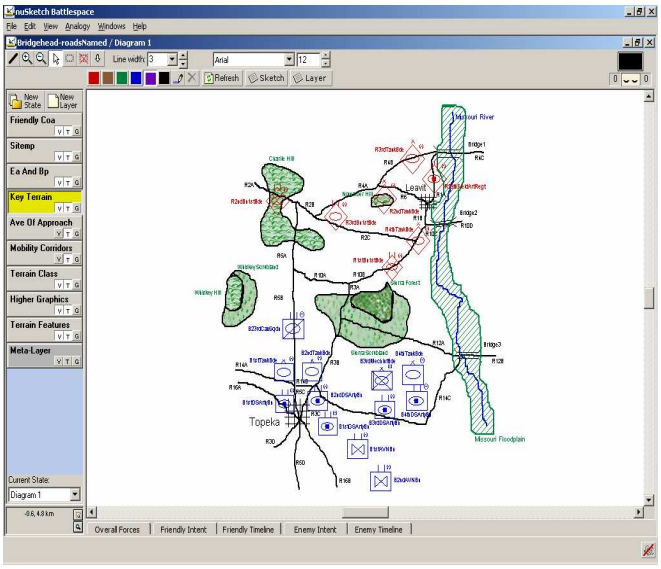


Figure 3: nuSketch COA authoring interface

NuSketch elements have a precise declarative semantics that is reflected in the SHAKEN component library ontology. Once the COA is specified in nuSketch, it is translated to a SHAKEN concept map (CMAP). The translator maps terms in the nuSketch ontology to the corresponding terms in the SHAKEN component library. In some cases, the knowledge is processed to resolve ontological mismatches; for instance, the task timing information in nuSketch is based on the quantitative start and end times, whereas SHAKEN relies on qualitative ordering information among tasks; therefore, the translator processes the quantitative information to derive the necessary qualitative ordering.

As expected, the experts want the interface to be as easy and quick to use as their regular pen-and-paper way of doing things. The primary obstacle to achieving this was to find a suitable combination of sketching gestures, and a layout of windows that would enable rapid authoring of the COA. Currently, it takes 1 to 2 hours to author a COA. The SMEs would like to be able to do it within 15 minutes.

Critiquing Knowledge

Critiquing relies on both domain-independent and specialized knowledge. Domain-independent knowledge is leveraged as domain-specific terms are created, by specializing domain-independent terms. We will primarily discuss here the domain-specific critiquing knowledge.

Two kinds of domain-dependent critiquing knowledge were needed: (1) necessary and sufficient slot values of concepts, and (2) critiquing rules. We now consider in more detail how each was entered.

Necessary properties of concepts: The SHAKEN graphical interface is the primary means used to create the domain-specific concepts from domain-independent ones. For example, for each kind of terrain, we encoded its trafficability for each kind of unit. For each unit, we encoded the equipment it possesses, and its combat power. For each military task, we encoded how much relative combat power is generally thought to be sufficient to effectively perform this task. The tasks are encoded using a STRIPS-like language used by many AI planners [4].

As a concrete example, Figure 4 shows the representation of the concept of *Rolling-Hills*. This concept map indicates that rolling hills offer relatively unrestricted movement for armor and infantry units. See [8] for a description of how logical axioms are synthesized from graphs such as this.

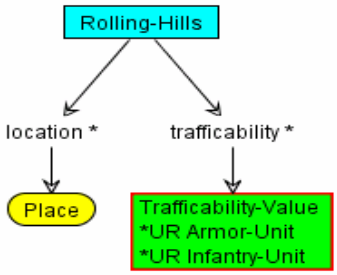


Figure 4: Trafficability definition for *Rolling Hills*

Sufficient properties of concepts: For many concepts, it is possible to define both necessary and sufficient properties. For example, if *Blue-Military-Unit* represents the class of all friendly units, then any military unit whose allegiance is *Blue* is a member of this class. A domain expert specifies the sufficient properties of a concept by annotating the graph representing the necessary properties.

The most common application of sufficient properties was to create subclasses of actions representing a specific situation, indicating a special case. For example, the required relative combat power ratio for the most general case of each military action is built into the system. However, the actual relative combat power ratio depends on the specifics of the situation. For instance, a ratio of 3 is normally desired for a general attack, but when an aviation unit attacks an armor unit, a combat power ratio of 0.5 is adequate. When a commander authors a COA, he may use the general attack action vocabulary. But, if the knowledge base includes a subclass of the attack action whose sufficient properties are that the agent is an aviation unit, and the object is an armor unit, its lower relative combat power ratio will be used whenever such a situation arises. Figure 5 shows the concept map for such a class. See [16] for more details on entering special cases of actions.

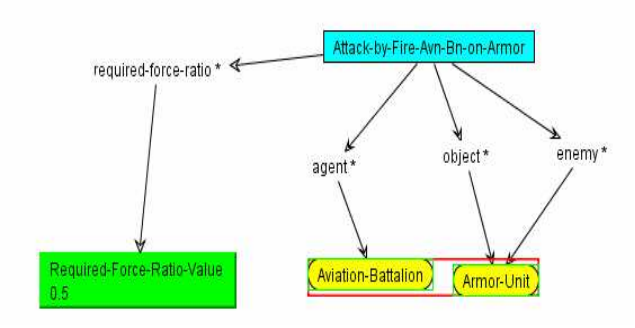


Figure 5: A special case of the *Attack* action. The nodes grouped in a box indicate sufficient properties.

Critiquing rules: We devised a special kind of rule, called a *pattern*, where the antecedent represents a collection of assertions pertaining to the situation being critiqued, and the consequent is a critiquing score on some critiquing dimension. Figure 6 shows an example pattern that rates a COA as good if some forces are kept in the reserve. The portion of the graph linked to the root with the *has-pattern* relation indicates an antecedent, and the portion linked using *critique-score* indicates the consequent of the rule.

Critique scores can be positive or negative, and a single pattern can apply to more than one critiquing dimension. Critiquing dimensions for COA patterns include such concepts as Risk, Casualties, Maneuver Effectiveness, Command and Control, Terrain Use, Preparedness for Enemy Response, Simplicity, Resource Use, and Synchronization. Applying these rules, organized by the critiquing dimensions, gives a direct rating of a COA.

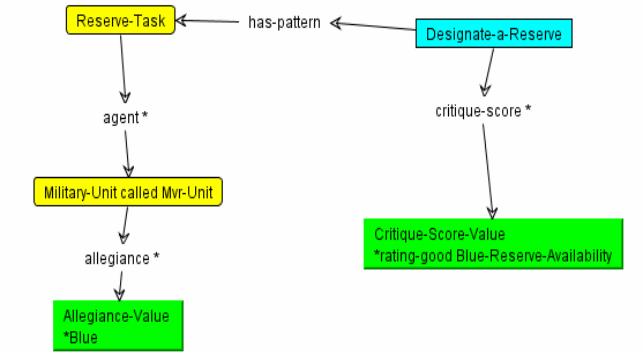
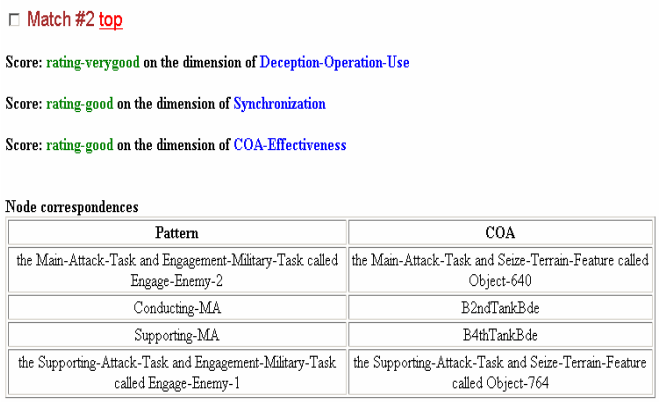


Figure 6: A pattern indicating that allocating a reserve is good for *Blue-Reserve-Availability*

Exercising Critiquing Knowledge

SHAKEN currently supports three different kinds of critiquing: declarative inference, normative simulation, and empirical simulation. (SHAKEN's analogical reasoning capabilities can also be used for critiquing [10], but this is not covered in the present paper.)

Critiquing by declarative inference: COA critiquing by declarative inference systematically finds and applies all applicable COA patterns and assigns them a score. The key technical challenge in matching patterns against a COA is that matches may not be syntactically exact. Therefore, we built a utility that can compute matches modulo a set of transformations. For example, we may know from the COA that a Blue force is in a city; we may also have a pattern saying that if an armor unit is in a city, it is poor for security of that unit (unless it is accompanied by infantry that can protect tanks in narrow streets and alleys from short-range antitank weapons). The pattern matcher will match the COA and the pattern, noticing that the Blue force has an armor unit that is in the same location. The pattern matcher contains a few hundred such transformations.



Match #2 top

Score: rating-verygood on the dimension of Deception-Operation-Use

Score: rating-good on the dimension of Synchronization

Score: rating-good on the dimension of COA-Effectiveness

Node correspondences

Pattern	COA
the Main-Attack-Task and Engagement-Military-Task called Engage-Enemy-2	the Main-Attack-Task and Seize-Terrain-Feature called Object-640
Conducting-MA	B2ndTankBde
Supporting-MA	B4thTankBde
the Supporting-Attack-Task and Engagement-Military-Task called Engage-Enemy-1	the Supporting-Attack-Task and Seize-Terrain-Feature called Object-764

Figure 7: A report from critiquing by patterns

Figure 7 shows an example report generated by matching patterns, as presented by the SHAKEN interface. The top of the report indicates the critiquing scores. The COA being evaluated has a score of *Very Good* on the dimension of deception. The table that follows indicates which nodes in the pattern matched which nodes in the COA. For example, *B2ndTankBde* conducts the main attack, and *B4thTankBde* conducts the supporting attack.

Critiquing by normative simulation: Normative simulation critiques a COA by executing each step. It relies on the KM situation mechanism, and executes each step based on its effects (add/delete lists). It analyzes dependencies between conditions and effects, checking that the required conditions for each step are met when the step is supposed to take place, and that the expected effects of the overall process are, in fact, obtained. It also checks how different steps are related to each other, including their temporal ordering and causal relationships. The simulation reports possible errors and presents them as

critiques. For instance, for each step in the COA, normative simulation computes the net relative combat power available, and compares it against the required relative combat power ratios already encoded in the system.

Figure 8 shows an example normative simulation report. In this case, one of the preconditions of a military action has failed: the given combat power ratio is not high enough to perform the given task. The net relative combat power of a military unit is computed based on the combat power of its subunits. The explanation section of the report shows in detail how the combat power was computed by combining various pieces of information, including unit equipment, default combat power, and remaining unit strength, through multiple COA steps. The user can check this explanation to see why the condition failed.

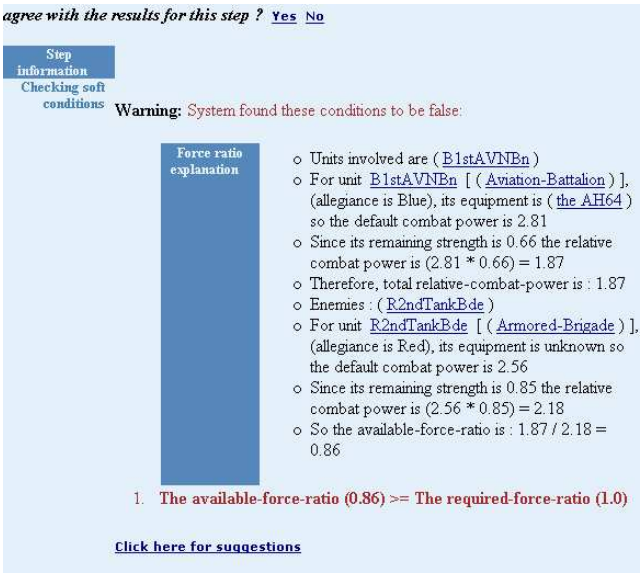


Figure 8: COA critiquing by normative simulation

The combat power numbers are dynamic, and take into account how the various units undergo attrition over a period of time. The action is flagged if the actual relative combat power during an action is less than the required relative combat power. Even when the combat power exceeds what is required, the commander can use the report information to check that all the decisive points have overwhelming relative combat power ratios.

In this instance, an SME added a special case of the Attack-by-Fire action to account for this kind of situation (i.e., when an aviation battalion attacks an armored unit, a combat power ratio of 0.5 is enough). Once this special case was added, the precondition was satisfied.

Critiquing by empirical simulation: Empirical and normative simulation complement each other in SHAKEN. Simulation is used to capture complex dynamics in the COA, and to explicitly model uncertainty. For empirical simulation, SHAKEN uses the Capture the Flag (CtF) tool [1], based on the AFS abstract physics-based model of division-level engagements described earlier (see Figure 2). Once a nuSketch COA is translated to CtF, Monte Carlo simulation is performed, running the COA multiple times until statistically significant results are obtained. The data from these trials is summarized in HTML reports, showing combat power ratios and graphical snapshots of critical events (e.g., engagements) during the simulated runs.

Figure 9 shows the combat power ratio graph produced for a particular engagement during a single simulation run. The ratio increases as the Blue side gains dominance over time, indicating a Blue army victory. A chief strength of empirical simulation is unexpectedly simple: SMEs can watch their COAs unfold visually, and can immediately see flaws and strengths. The results are analyzed to construct a qualitative representation of the space of outcomes, explicitly identifying critical points.

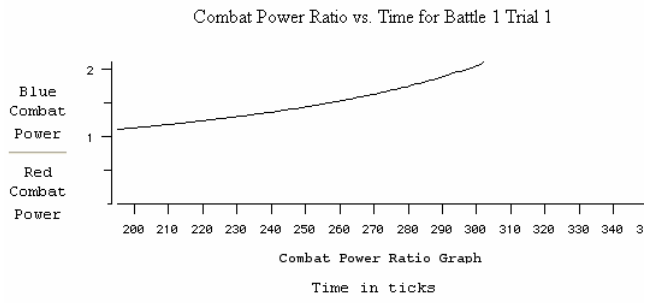


Figure 9: Output from empirical CtF simulation

Evaluation

We evaluated the system with the help of two domain experts, both of whom were retired Army officers. One had served at the rank of lieutenant general, and the other as an intelligence officer. The objective of the evaluation was twofold: to assess how effectively the knowledge acquisition capabilities of SHAKEN would work for domain experts with no training in formal knowledge representation, and to test the performance of the resulting knowledge base on the COA critiquing task.

The evaluation was conducted over 15 days. During the first 7 days, we provided hands-on training to the two subject matter experts, using an example critiquing task. The SMEs were then given a new task, in the form of a COA problem statement and its solution, expressed in textual form, and were asked to address it using the system. The SMEs were asked to encode the textual description in SHAKEN. They then authored critiquing knowledge, independent of the COAs, and used it to critique them.

Before encoding a COA, the SMEs produced a manual critique for it, to serve as a guideline for evaluating the ultimate critique to be produced by the system. Authoring the critiquing knowledge was an iterative task: the knowledge was successively refined based on the system critique, and how it differed from the manual critique.

Over the 15-day period, the SMEs authored three different COAs and 60 pieces of critiquing knowledge. The critiquing knowledge included patterns and special cases of actions. Below, we present the textual description of a few patterns authored by the SMEs during evaluation. The critiquing dimensions are shown in bold font:

If a COA secures a piece of terrain narrower than 50 meters, it makes good ***use of terrain****.*

If the supporting attack occurs before the main attack, it is good for COA ***effectiveness, mission accomplishment****, and* ***synchronization****.*

If an armored unit attacks a mechanized infantry unit outside a city, it is good for ***enemy maneuver engagement.***

The antecedent encodes the condition under which the pattern applies, and often includes spatial information such as terrain or unit location. In some cases, the antecedent can include negation, for example, the location of a unit not being in a city. Let us now consider two examples of special cases of actions, where the bold text represents the sufficient property of the special case:

When ***an aviation unit attacks an artillery unit,*** *it is sufficient to have a combat power ratio of 0.3.*

While ***seizing a bridge****, it is sufficient to have a combat power ratio of 0.3.*

These example patterns and special cases of actions show that SMEs with very little training in knowledge representation were able to author nontrivial pieces of critiquing knowledge. In particular, the first-order logic formalization of this knowledge, synthesized automatically from the graphs by SHAKEN, includes quantified variables, implications, negation, and, in the case of special cases of actions, concept definitions (bi-directional implications). These formal structures are clearly beyond anything that the SMEs could encode directly. In addition, through the constraints imposed by the graphical interface (e.g., guiding the SME to select concepts from the existing ontology, restricting the choices of relations to only semantically valid ones), the SMEs formalized their knowledge in conformance with SHAKEN's underlying ontology. This illustrates the key achievement of this work, namely, a significant enhancement of the SME's ability to articulate formal knowledge, in a way consistent with, and building upon, the preexisting knowledge in the system.

We tested the empirical simulation on the COAs authored by the SMEs. Monte Carlo summaries of mass lost and goals achieved over multiple simulations showed clear differences between these COAs. In addition, the COAs that we felt were most dangerous had the greatest amount of variance in their outcome. This highlights one of empirical simulation's greatest strengths: the ability to go beyond static analysis and focus instead on the dynamics of multiple concurrent processes.

Despite these achievements, we encountered several limitations. The most significant problem is to translate natural but informal domain concepts (e.g., "sufficient force", "flank", "contour", "overwhelm") into a computable form (e.g., in terms of coordinates and distances), a prerequisite for machine reasoning about the domain. While SHAKEN provides good support for entering formal knowledge once that conceptual translation is made, it provides little help with the translation in the first place. This turned out to be the most notable challenge for the SMEs. It is exacerbated in the COA domain, where many important concepts are spatial in nature, but particularly difficult to pin down precisely in formal terms.

Second, although the interface helps SMEs enter knowledge in terms of the existing ontology, there is still potential for SMEs to make mistakes. For example, they sometimes used negation in a way that differed from their intent, without realizing that the semantics of what they encoded was subtly different (e.g., one SME encoded "an attack not on a city is good", intending to encode "no attack on a city is good"). More proactive checking and validation of SME inputs would help identify and correct such errors.

As additional evaluation data, at the end of the 15-day period we compared the SHAKEN critiques produced using an SME's formally encoded knowledge with the manual critiques written by the same SME. Our goal was to check that the SME's encoded knowledge was to some extent "reasonable" compared with his ideal solution (the manual critique), that is, to check that the SME's rules were not simply "formal nonsense". The SMEs were asked to assign a correctness score on a five-point scale (-2 to +2) to the results produced by SHAKEN using their encoded knowledge. A score was given to each critiquing dimension that the SME considered relevant to the particular COA.

Of the 16 relevant critiquing dimensions for one of the representative COAs, the system critique received a score of +2 for 8 of the dimensions; for 3, a score of +1; for 4, a score of –1; and for 1, a score of -2. Although many other factors influence these scores (e.g., the inherent knowledge representation and reasoning capacity of SHAKEN itself), the results indicate that the SME was able to enter at least some of his knowledge with a reasonable degree of accuracy and fidelity.

The SMEs' overall assessment was that a COA analysis capability such as the one we tested could ultimately be very useful in solving operational problems:

The software can work through tedious details and double-check all potential COAs, especially when the commanders are tired, under pressure, and under time constraints.

Although our goal is to break new ground in knowledge acquisition technology, rather than to specifically critique COAs, it is nevertheless interesting to consider what it would take for the COA-critiquing application of SHAKEN, using SME-entered knowledge, to reach a sufficiently mature level for deployment. The technology requires numerous enhancements before it comes close to being deployable. For example, a library of a few hundred patterns and special cases of action will have to be built before the system starts producing non-obvious critiques that add value to what a commander can quickly determine with a visual inspection of a COA. One way to drive such a knowledge base construction is to work with a sizable collection of case studies [23] that will provide concrete test cases, a well-defined scope for knowledge entry, and clear performance criteria. The detail captured in the normative simulation can also be improved, giving special attention to simulating concurrent events.

Related Work

In previous work, we developed an extensive ontology of plan evaluation and plan critiquing [5]. In another previous study, we evaluated nuSketch as a COA authoring tool, and demonstrated that COAs authored using nuSketch were comparable in quality to ones authored with more traditional methods [22].

In the present work, the main innovations are: (a) using the plan critiquing ontology in conjunction with normative simulation; (b) acquiring critiquing knowledge in the form of patterns and necessary and sufficient conditions for actions; and (c) showing that the system can exhibit some level of COA critiquing competence, through declarative inference, normative simulation, and empirical simulation.

There has been significant work in building interactive plan authoring environments [20], but it has not addressed the specific problem of COA critiquing. The use of patterns for COA critiquing was demonstrated in [11], which let experts select subsets of a COA sketch to generate critiques that could be subsequently applied via analogy or as rules. However, that system only used information explicitly represented in the sketch, whereas a broader range of knowledge can be used in SHAKEN patterns.

Future Work

Work is under way to address many of the limitations identified in the previous section. For example, we are making extensions to nuSketch to support richer COA descriptions. Similarly, we are implementing the normative simulation of concurrent events.

We are also developing a suite of capabilities that will let SHAKEN users enter, organize, and retrieve knowledge using English. These capabilities make use of the START [14] and Omnibase [15] systems. To perform knowledge entry, the user enters a sentence or phrase, which is parsed into a concept map representation similar to that used within SHAKEN. Through an interactive dialog between the user and the system, this concept map is refined into a SHAKEN concept map, which is added to SHAKEN's knowledge base. Using a similar approach, English questions are translated into concept map patterns, which are then used to identify matching concepts within SHAKEN's knowledge base.

Summary

We presented the application of a general-purpose knowledge-based system, SHAKEN, to the specific task of military Course of Action (COA) analysis. We showed how SHAKEN can capture and reuse expert knowledge for COA critiquing, and produce a high-level assessment of a COA through declarative inference and simulation. The system has been used and evaluated by domain experts. The generality of the approach makes it applicable to knowledge capture for task analysis in other domains.

Acknowledgments

We thank Commander Dennis Quinn and Lt. Gen. Len Wishart for serving as the domain experts for the evaluation, and Murray Burke for his encouragement and support for this work. This research was supported by DARPA's Rapid Knowledge Formation project under contract N66001-00-C-8018.

References

1. Atkin, M., G.W. King, D. Westbrook, B. Heeringa, A. Hannon, and P. Cohen. *SPT: Hierarchical Agent Control: A Framework for Defining Agent Behavior*. In *Fifth Intl. Conf. on Autonomous Agents*, p. 452-432, 2000.
2. Atkin, M.S., D.L. Westbrook, P.R. Cohen, and G.D. Jorstad. *AFS and HAC: Domain-General Agent Simulation and Control*. In *Workshop on Software Tools for Developing Agents, AAAI-98*, p. 89-95, 1998.
3. Barker, K., B. Porter, and P. Clark. *A Library of Generic Components for Composing Knowledge Bases*. In *International Conference of Knowledge Capture*, 2002.
4. Blythe, J. *SHAKEN Action Description Language*. Technical Report, Information Sciences Institute, University of Southern California, 2002.
5. Blythe, J. and Y. Gil. *A Problem-Solving Method for Plan Evaluation and Critiquing*. In *Intl. Knowledge Acquisition Workshop*. Banff, 1999.
6. Clark, P., K. Barker, B. Porter, A. Souther, V. Chaudhri, S. Mishra, J. Thomere, J. Blythe, J. Kim, P. Hayes, K. Forbus,

and S. Nicholson. *A Modified Template-Based Approach to Question-Answering from Knowledge Bases*. Technical Report, SRI International, Menlo Park, CA, 2002.
7. Clark, P. and B. Porter. *KM -- The Knowledge Machine User Manual*. Technical Report, U. of Texas at Austin, 1999.
8. Clark, P., J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, and T. Reicherzer. *Knowledge Entry as Graphical Assembly of Components*. In *Intl. Conf. on Knowledge Capture*, 2001.
9. Forbus, K., R. Ferguson, and D. Gentner. *Incremental Structure Mapping*. In *Proc. Cognitive Science Society*, 1994.
10. Forbus, K., R. Ferguson, and J. Usher. *Towards a Computational Model of Sketching*. In *Intelligent User Interfaces Conference*. Santa Fe, New Mexico, 2001.
11. Forbus, K., T. Mostek, and R. Ferguson. *An Analogy Ontology for Integrating Analogical Processing and First-Principles Reasoning*. In *IAAI-02*, 2002.
12. Forbus, K. and J. Usher. *Sketching for Knowledge Capture: A Progress Report*. In *Intelligent User Interfaces*. 2002.
13. Forbus, K., J. Usher, and V. Chapman. *Sketching for Military Courses of Action Diagrams*. In *Proceedings of Intelligent User Interfaces Conference*. Miami, FL, 2003.
14. Katz, B. *Annotating the World-Wide Web using Natural Language*. In *5th RIAO Conference on Computer Assisted Information Searching on the Internet*, 1997.
15. Katz, B., S. Felshin, D. Yuret, A. Abrahim, J. Lin, G. Marton, A.J. McFarland, and B. Temelkuran. *Omnibase: Uniform Access to Heterogeneous Data for Question Answering*. In *7th International Workshop on Applications of Natural Language to Information Systems*, 2002.
16. Kim, J. and J. Blythe. *Supporting Plan Authoring and Analysis*. In *Intelligent User Interfaces*. Miami, FL, 2003.
17. Kim, J. and Y. Gil. *Knowledge Analysis on Process Models*. In *17th Intl. Joint Conf. on Artificial Intelligence (IJCAI-2001)*, p. 935-942, 2001.
18. King, G.W., M.S. Atkin, and D. Westbrook. *Tapir: The Evolution of an Agent Control Language*. In *First Conference on Autonomous Agents and Multiagent Systems*, 2002.
19. Mishra, S., A. Rodriguez, M. Eriksen, V. Chaudhri, J. Lowrance, K. Murray, and J. Thomere. *Lightweight solutions for user interfaces over the WWW*. In *International Lisp Conference*. San Francisco, CA, 2002.
20. Myers, K. *Strategic Advice for Hierarchical Planners*. In *Intl. Conf. on Knowledge Representation and Reasoning*, p. 112-123, 1996.
21. Nicholson, S. and K. Forbus. *Answering Comparison Questions in SHAKEN: A Progress Report*. In *Spring Symposium on Mining Answers from Text and Knowledge Bases*. Stanford, CA, 2002: AAAI.
22. Rasch, R., A. Kott, and K.D. Forbus. *AI on the Battlefield: An Experimental Exploration*. In *Innovative Applications of Artificial Intelligence*. Edmonton, Canada, 2002.
23. Schmitt, M.J.F., USMCR, *Mastering Tactics: A Tactical Decision Games Workbook*. Marine Corps Gazette. 1994, Quantico, VA: Marine Corps Association.