

Automating the Acquisition of Tactical Knowledge for Military Missions

Avelino J. Gonzalez

Intelligent Systems Laboratory
School of Electrical Engineering
and Computer Science
University of Central Florida
Orlando, FL 32816
gonzalez@ucf.edu

Jose Castro

Ingeniería de Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
jcastro@ucf.edu

William Gerber

Dynamics Research Corporation
Orlando, FL 32817
bill.gerber@ieee.org

It is widely accepted that acquisition of the knowledge behind military tactics has been one limiting factor in the development of computer generated forces (CGF) for training simulations. This has been addressed by several researchers with varying degrees of success. A system capable of building a knowledge base directly from a dialogue with a subject-matter expert (SME) could significantly reduce the human effort involved in capturing the knowledge and representing it directly in the modeling language. Because of its highly modular and hierarchical nature, the context-based reasoning (CxBR) modeling paradigm lends itself very well to facilitating the knowledge acquisition process for tactical behaviors. This paper describes an investigation into using CxBR as the foundation for a system that creates a (partial) model of tactical behavior through an interactive process with an SME. Through a sequence of queries from the system, the SME is progressively asked to provide details about the contexts that compose the context-based model of the expert's tactical know-how. A prototype was built and evaluated. A comparison to the effort taken to manually develop a knowledge base is reported. We use the simulation of a non-trivial maritime military confrontation as the benchmark for the comparisons.

Keywords: Automated knowledge acquisition, knowledge engineering facilitators, knowledge engineering tools

1. Introduction

Over the past decade or so, research in computer generated forces (CGF) has focused on better ways to represent tactical human behavior for military training simulations. CGF agents can act in place of teammates or enemy vis-à-vis specific missions, thereby reducing the need for human players or support staff. Many advances in representation of tactical knowledge have been made in the recent past with systems such as ACT-

R [1], COGNET [2], CxBR [3], SOAR [4, 5], and several others. Nevertheless, it has become increasingly clear that the effort required to build tactical CGF agents is both large and costly.

1.1 Automated Knowledge Acquisition

The general concept of facilitating the acquisition of the knowledge for intelligent systems has been a goal of AI researchers for several years. Some of the early work in automated knowledge acquisition tools revolved around weak models used for heuristic classification. Early weak-model systems such as Planet [6], ETS [7],

AQUINAS [8], and Auto-Intelligence [9] employed repertory grids to store the knowledge as well as to guide the interaction with the subject-matter expert (SME). They were specifically used for classification and/or diagnostic tasks. Strong model systems such as MORE [10] refined domain models through a dialogue with the SME. However, most of the early systems were limited to classification or diagnostic tasks. SALT [11] was the first knowledge acquisition system that addressed tasks of synthesis such as configuration. Recent advances in this general area of knowledge acquisition include ontology building tools like Ontolingua [12], Protégé [13], and concept maps [14]. Other tools such as KanaIQ (www.kana.com) and Protos [15] are based on machine learning, specifically exemplar learning.

In the early 1990s, method-based knowledge acquisition systems began to appear in the literature. These strong-model systems approached the knowledge acquisition problem by building specific tools that addressed the specific structure of the task faced by the system being built. Taql [16] is one of these method-based systems, although the authors claim that it is more general than are other task-specific method-based systems. Another system worth mentioning is EMEd [17]. Through interaction with the SME, EMEd automatically infers the interrelationships among the concepts elicited. The authors' tests indicate that significant time savings are achieved through the use of this tool.

1.2 Tactical Behavior

While the advances in automating the knowledge acquisition process have been steady as well as impressive, none of the above approaches specifically addresses the acquisition of *tactical* knowledge. Tactical knowledge is categorically different from classification knowledge and from general problem-solving knowledge because it is inherently temporal. Shutte [18] states that tactical behaviors are "...near-term and dynamic activities..." Verbs associated by his test subjects with tactical behavior included *respond*, *act*, *react*, and *do*, among others. Therefore, tactical behavior implies some activity(ies) executed over time. Classification knowledge, on the other hand, merely results in a decision about class membership. General problem-solving knowledge may not always result in some activity (e.g., diagnostics, design, planning, and scheduling). While mostly activity-based but partly goal-based, tactical knowledge requires the full range of knowledge levels: from the low-level motor skill knowledge (e.g., how to keep a car on the center of the lane) to the high-level course-of-action selection knowledge (i.e., whether to take the freeway or the back roads).

Tactical knowledge can be defined as that knowledge which permits an agent to act intelligently and realistically in light of the situation faced. Thorndike [19] asserts that tactical behavior includes tasks that require 1) assessment of the situation at hand, 2) selection of a plan to most properly address the present situation, and 3) execution of that plan. We define it as

The *continual* and *dynamic process* of decision making by a performing agent (human or otherwise) that interacts with its environment while attempting to carry out a mission in the environment. Decisions are made at *several levels* during the execution of the mission.

Tactical behavior involves the concept of situational awareness, or SA [20]. Endsley [21] defines SA as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future." Endsley [20] defines three levels of SA: 1) perceiving the event or aspect of the environment, 2) understanding the implications of the event/aspect vis-à-vis the mission goals, and 3) being able to predict what will happen as a result of these events/aspects of the environment. In our example, situational awareness would be learning of an accident on the freeway that a driver is in or is headed toward; realizing that the accident may be blocking traffic on the freeway; and projecting that if she takes (or stays on) the freeway, her movement on that freeway may become blocked, thus preventing her from reaching her destination. SA would lead her to deduce that taking back roads may be a better choice.

In military confrontations, the selection of the appropriate plan of action to address the current situation is typically guided by military doctrine. Such doctrine is taught to enlisted personnel and officers and is described in several military publications (e.g., field manuals). U.S. military doctrine, however, unlike that of some other countries, permits a certain amount of freedom of choice to the combatant, as it would be impossible for the doctrine to cover in detail all possible situations likely to be faced in combat. This unpublished knowledge can be considered to be heuristic in nature and is generally learned through an individual's own experience.

Humans also use tactical knowledge in non-military competitive settings such as team sports, and for everyday tasks like driving an automobile or navigating a boat. Such knowledge is also partly published (e.g., rules of baseball, traffic laws, nautical rules of the road) and partly exists as heuristics learned through training or experience. Thus, appropriate tactical behavior in accordance with accepted practice is partly well defined in the published rules or doctrine and partly heuristics based on an individual's

experience and expertise. The basis of our approach to concisely and efficiently represent the knowledge of a tactical CGF agent is called *context-based reasoning* (CxBR). CxBR considers both of these components in how it represents tactical knowledge. We discuss CxBR later in this paper, as it is central to our approach to automated knowledge acquisition.

1.3 Acquisition of Tactical Knowledge

Some research work that addresses the problem of acquiring tactical knowledge has been reported in the literature. Randolph [22] describes two methodologies called the *personal knowledge acquisition process* and the *team knowledge acquisition process* for acquiring tactical knowledge in domains to be simulated. However, it is a manual technique. Delugach and Skipper [23] describe an interactive approach to acquire tactical knowledge based on repertory grids that are then converted into conceptual graphs. They refer to them as *tracked repertory grids*. The authors claim it facilitates tactical knowledge acquisition for CGFs but provide no support for that claim. Other recent approaches involve learning directly by the agent, generally by observation [24–31]. However, such machine learning strategies are still in various stages of maturity and do not yet provide near-term relief for the acute problem of acquiring and representing large volumes of tactical knowledge, even for relatively simple missions.

In this article, we describe our approach to semi-automatically building tactical agents in the form of computer generated forces. A tool embodying this approach could significantly reduce the effort required to build these agents. Our approach is based on the inherently modular and highly structured nature of CxBR for representing tactical human behavior. Its structured form and hierarchical organization lends itself very well to an automated (or semi-automated) query system. We call this process *context-based interactive tactical knowledge acquisition* (CITKA). We claim that CITKA is a strong model, method-based approach that is domain-independent, but only as long as the nature of the knowledge is tactical. We describe this process in section 3 below. Prior to that, a brief discussion of CxBR and of the features that make it an excellent medium for representing and (semi-)automatically acquiring tactical knowledge is presented.

2. Context-Based Knowledge

The hypothesis of our investigation is that CxBR is well suited for building a query system for (semi-)automatically eliciting and collecting tactical knowledge from an SME, with minimal (but finite) effort from a

knowledge engineer (KE). Henninger [32] suggested this in 1997 and showed potential savings in a simple application. While important, Henninger's prototype lacked an integrated control mechanism to direct the interaction with the SME. Furthermore, its user interface was rather primitive, thereby prohibiting realistic testing with an SME. The work described in this paper is motivated in part by Henninger's ideas. However, it goes significantly beyond her work by having a rule-controlled dialogue and a more suitable user interface that permits a rigorous evaluation of the prototype with an SME.

We assert that by interacting with the SMEs in a way that simulates the queries posed by a human interviewer, and managing their responses, a context-based model of human tactical behavior can be built *semi-automatically*. We qualify the process as being "semi-automatic" because we believe that this approach will never achieve the development of a complete model, untouched by human hands; a KE will ultimately have to step in and finish the system being developed. Our objective is to minimize this intervention by the KE. We begin by briefly discussing CxBR. See Gonzalez and Ahlers [3] for a thorough discussion on this subject.

2.1 Context-Based Reasoning

CxBR is a modeling paradigm specifically designed to represent human behavior in tactical situations. CxBR is modular and hierarchically structured, yet flexible enough to permit great leeway in how the agent is built. It is based on the idea that when executing a mission, an autonomous agent sequentially experiences several different situations. Each situation will require certain skills, decisions, and actions in order to successfully manage it. Furthermore, situations evolve throughout a mission, often shifting quite abruptly, other times gradually. To successfully complete the mission, the agent must have the skills required to "navigate" each of the tactical situations and must recognize when the situation has changed. These tactical situations can be likened to *contexts*. Therefore, the three basic tenets of CxBR are:

- 1) A tactical situation calls for a set of skills (actions and procedures) and decisions that are relevant to the current situation;
- 2) As a mission evolves, a transition to another set of actions and procedures may be required to address a newly emerging situation; and
- 3) Things that can happen while the agent is in the current situation are limited by the current situation itself.

CxBR encapsulates knowledge about appropriate

actions and procedures into hierarchically organized *contexts*. This hierarchy covers the entire range of actions, from high-level decisions to low-level actions. The context hierarchy is in some ways similar to a task hierarchy. However, contexts encompass more than tasks because they are responsible for monitoring the environment and knowing when they are no longer applicable. This feature is called the context *transition*, and each context will contain its own *transition criteria*.

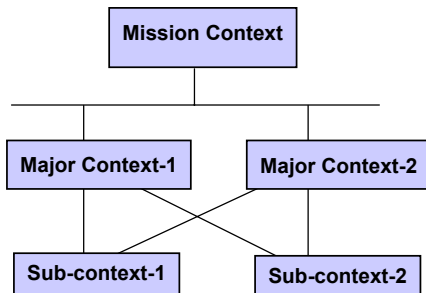


Figure 1. Context hierarchy

Furthermore, the top-level context (the Mission) is not a task but a definition of a complex mission. A simple context hierarchy is depicted in Figure 1.

Mission Contexts define the mission to be undertaken by the agent. While it does not control the agent per se, the Mission Context defines the scope of the mission, its goals, the plan for its execution, and the constraints imposed (time, weather, rules of engagement, etc). The *Major Context* is the primary control element for the agent. Each (of several) Major Context defines a situation to be experienced during the mission. A Major Context contains functions, rules, and links to *Sub-contexts*. Knowledge about what to do in a context (action knowledge) is typically expressed as functions, rules, and methods. However, it can also be expressed as traditional mathematical equations, neural networks, cases, constraints, fuzzy logic, and other such programming paradigms. This gives CxBR great flexibility in representation.

Major Contexts also contain a list of *compatible-next-Major-Contexts*. These are contexts deemed realistic for the agent to *activate* after the current active context. Compatibility of contexts for a particular transition is currently predetermined by the SME and, secondarily, by the KE. Identification of a new situation is simplified because it is possible for an agent to transition to only a limited number of all possible situations while it is under the control of the currently-active context.

Sub-contexts are lower-level, narrowly focused, yet complex actions performed by the Major Context. Sub-contexts are activated by rules within the *active* Major Context (the one controlling the agent at any

one time). Sub-contexts deactivate themselves upon completion of their actions, passing control back to the Major Context that activated them.

Building models of tactical human behavior in CxBR is a highly structured process. This has the potential of facilitating the knowledge acquisition process significantly. Specifically, the following knowledge must be captured and defined in a CxBR model of tactical human behavior.

- 1) The Mission Context must be identified, and a value set for its applicable attributes. These include name of the mission, description, constraints, and objective(s) to be achieved in the mission. These should all be known by the SME.
- 2) The Major Contexts and Sub-contexts to be used must be identified and labeled by the SME.
- 3) For each Major Context and Sub-context identified above, the methods, procedures, and functions required for controlling the agent while under the appropriate Major Context or Sub-context must be defined, specified, coded, and incorporated within the appropriate context class. These are called the context's *control actions*, driven by the *action knowledge*.
- 4) The rules incorporating the criteria for context transitions must be identified, specified, written, and integrated within the appropriate context class. These are called *sentinel rules*. While other paradigms could be used for this purpose (e.g., neural networks), we will assume here that rules are being used.
- 5) The *objects* involved in the mission must be identified, and their characteristics must be specified. Objects are actors, friendly or enemy, that will play a role in the simulation. For example, if the mission is for a platoon of tanks, each individual tank's maximum speed, turning radius, fuel capacity, weapons load, etc., must be defined. Also included here must be any enemy or friendly elements with which our agent must contend. The SME should know what the relevant objects are for the particular mission contemplated, as well as their required attributes.
- 6) Helping functions must be identified, specified, and defined. Examples of helping functions include finding the distance between two points and selecting the heading required for reaching a particular waypoint.

While experience-based heuristics can be reflected in the control action knowledge within a context, the tacit heuristics that are so often difficult to elicit from SMEs are typically represented as sentinel rules. The close associations that the sentinel rules or heuristics

embedded in the control action have with their relevant context virtually eliminate the need to express the exceptions to these heuristics. This also facilitates the knowledge acquisition process.

A brief example of a context-based agent for a submarine patrol mission is provided below.

2.2 Brief Example of a Context-Based Agent of a Submarine on a Patrol Mission

A tactically intelligent submarine agent is to be built and executed in CxBR. The mission of our agent is to patrol a sector of ocean and monitor enemy vessels within it. We shall call this Mission PATROL. If an enemy submarine is detected and identified, the agent is to follow it until it is told to break contact and return to base, or given orders to attack the enemy submarine. If attacked first, our agent is to defend itself and counterattack the aggressor.

The model that controls the agent in a PATROL mission (a Mission Context) can be composed of Major Contexts representing the following actions:

- 1) Transiting to the assigned sector (**Transit-to-Sector**),
- 2) Patrolling the sector upon arrival (**Patrol-Sector**),
- 3) Following an enemy submarine when contact is made (**Track**), and
- 4) Returning to home port when mission ends (**Transit-Home**).

Each of these situations requires different skills and knowledge to execute successfully. A plan could be composed of these four Major Contexts, in sequence, with some criteria defining when to transition amongst them. For example, the transition between **Patrol-Sector** and **Track** would normally be indicated by the first sign of contact with an enemy submarine. Likewise, transition between **Transit-to-Sector** and **Patrol-Sector** would result from the agent reaching its designated sector. It would change its behavior from one of **Sprint-and-Drift** (a Sub-context within **Transit-to-Sector**) to moving stealthily at quiet speed and a depth below the thermocline (a function within the **Patrol-Sector** Major Context).

Other unplanned but possible situations could arise that must also be included. The enemy may attack, causing the agent to evade any weapons fired in its direction (**Evade**). The agent may be ordered to attack the submarine being tracked (**Attack**) or to break contact and escape (**Break-Contact**). Additionally, there may be an emergency Major Context activated when a collision with a moving or stationary object is imminent (**Avoid-Collision**).

The mission is described by the **PATROL** Mission Context (item 1 of the six items in the previous section).

Each of the above eight situations and corresponding actions represent Major Contexts (item 2). Each Major Context has control functions (items 3 and 6) and/or Sub-contexts (item 2) that permit them to successfully control the actions of the submarine agent when in their specific situation. They also contain the sentinel rules (item 4) that monitor the environment for signs that this Major Context is no longer applicable to the changing situation (e.g., enemy submarine detected while in **Patrol-Sector**). For example, a sentinel rule would state that “upon detection of enemy submarine, activate the **Track** Major Context.”

Lastly, the objects necessary to simulate the mission are defined (item 5). These could include friendly vessels nearby, enemy submarines, as well as large marine mammals that could confuse signals. Refer to Gonzalez and Ahlers [3] for details on CxBR. An abbreviated example of context-based specification for a submarine agent follows. It includes the Mission Context and one Major Context—the **Track** Major Context.

Mission Context: PATROL-SECTOR

Description: Patrol assigned sector of ocean to seek and track enemy submarines.

Initial location: King’s Bay submarine base

Goal destination: Sector “Charlie,” defined by four GPS coordinates

Objective: Seek and track enemy boats and report on their activities.

Constraints on the mission:

- 1) Do not attack unless attacked, or if specifically ordered to do so;
- 2) Performance of submarine agent (e.g., maximum depth, speed, turning radius, others);
- 3) Avoid physical contact with all other moving and stationary objects; and
- 4) Self-preservation is of primary priority.

Planned sequence of Major Contexts:

- 1) **Transit-to-Sector:** Move to sector “Charlie.”
- 2) **Patrol-Sector:** Seek enemy to find enemy in sector.
- 3) **Track:** Follow enemy submarine and report on its activity.
- 4) **Transit-Home:** Return to base upon completion of mission.

Reactive Major Contexts:

- 1) **Evade:** Take evasive action when attacked.
- 2) **Attack:** Position self, and fire weapons at enemy.

- 3) **Break-Contact:** Escape from enemy detection upon discovery by enemy.
- 4) **Avoid-Collision:** Take emergency action to avoid collision with fixed or moving object.

Track Major Context Definition

Action Function #1: Submerge to depth of enemy submarine. Maintain depth.

Action Function #2: Maintain constant distance to enemy submarine.

Helping Function #1: Steer. Change heading of submarine.

Helping Function #2: Accelerate. Increase speed of submarine gradually until goal speed is reached. Provide acceleration rate.

Helping Function #3: Decelerate. Decrease speed of submarine until goal speed is reached. Provide deceleration rate.

Sentinel Rule #1: If ordered to attack tracked vessel, activate **Attack** Major Context.

Sentinel Rule #2: If attacked, activate **Evade** Major Context.

Sentinel Rule #3: If detected by enemy submarine, activate **Break-Contact** Major Context.

Sentinel Rule #4: If rapidly nearing object, and time to collision is < 10 seconds, activate **Avoid-Collision** Major Context.

Sentinel Rule #5: Upon mission accomplishment, activate **Transit-Home** Major Context.

Relevant Sub-contexts: **Get-Behind**

Compatible-next-Major-Contexts: **Transit-Home, Attack, Evade, Avoid-Collision, Break-Contact**

Several intelligent tactical agent applications have been developed in CxBR. They range from military platforms (submarines, tanks) to civilian vehicles such as automobiles and commercial vessels; see Grejs [33], Brown [34], Gumus [35], and Proenza [36]. The early implementations were done using the CLIPS tool. CLIPS is an expert system development and execution environment developed by NASA JSC and is widely available. CLIPS served as an adequate vehicle for implementing CxBR in its early forms. In 1998, Norlander [37] developed a CxBR framework that provided an infrastructure optimized for representing human behavior in CxBR. This replaced the use of CLIPS as the implementation tool. Subsequently, Devero [38] proceeded to add the capability to incorporate rudimentary collaborative behavior between members of a team with a common goal into Norlander's CxBR framework. It was also used as the basis for representing behavior of entities that have

been degraded through either enemy fire or equipment malfunction [39].

The general concept of modeling human behavior through contexts has been investigated by several other researchers. Turner [40, 41], Brezillon [42], and Bass [44] have all independently developed context-based approaches to modeling human behavior.

It is clear from the above discussion that models in CxBR are highly structured. Developing these models is facilitated by this highly structured nature. CITKA takes advantage of this to elicit knowledge from SMEs in an organized fashion. Let us now see how CITKA works and discuss the concepts that underlie it.

3. The Agent Building Process

Yost [16] defines five basic components found in method-based knowledge acquisition systems: 1) a well-defined task type; 2) a "method" that can use knowledge to perform this task; 3) a representation of the methods and the knowledge; 4) a language with which to define the knowledge; and 5) a mapping from the language to the methods and knowledge and, ultimately, to the machine-readable representation. The main advantage of CITKA as a method-based knowledge acquisition tool is that it is very tightly tied with CxBR, the modeling paradigm that represents the knowledge and executes the resulting models. The CITKA process satisfies these components quite well. The well-defined task is that of a tactical nature. The method is the context execution and transition process. The contexts and the context hierarchy form the representation of the knowledge and the methods. While a language is not specifically defined in CxBR, we see this as an advantage rather than as a detriment, as the KE can incorporate whatever language best fits the application. CxBR merely provides an organization and high-level means of execution within which to incorporate this language. For example, one can use rules, functions, case-based reasoning, or almost any other computational paradigm within a context definition. This is unlike other systems such as SOAR and ACT-R, which strictly define the language used. The mapping is also present, as the CITKA process queries the SME to directly create contexts and sentinel rules. In effect, CITKA requires no mapping per se, as the SME is asked to directly create and populate the context objects and to provide the transition criteria. This direct "mapping" and use of the inherent organization of CxBR is what makes us believe in its superiority over other methods, especially for tactical behaviors; however, we admittedly have no empirical evidence to support this statement at the moment.

The process of building a model of tactical human behavior begins with a specification of the capabilities

and constraints of the tactical agent to be controlled by this model. This specification is mission specific, as one would expect the assets of a military task force to differ from mission to mission. Additionally, the enemy faced may also have widely varying capabilities. Once specified, the agent is developed by building the *context base*; that is, by defining the contexts, the action knowledge, the sentinel rules for transitions between contexts, and the necessary objects in the simulation—the six items in section 2.1 above. Once created, the context base for the agent is linked to the CxBR framework, and then to the simulation of choice for execution. The CxBR framework exercises the context base to achieve the desired behaviors, actions, and decisions by the agent. This knowledge is represented as contexts (objects) and functions in the context base. We assume here that the CxBR framework is already linked to the simulation of choice. Our objective is to 1) specify and 2) develop the context base. CITKA addresses these tasks.

CITKA has two basic functions: 1) to query the SME to elicit the tactical knowledge and 2) to build the context base using the SME's response. Additionally, it provides a KE with access to edit and enhance the context base. Therefore, the prototype system consists of four modules of independent but cooperating subsystems:

- 1) Knowledge engineering database backend;
- 2) Query rule base backend;
- 3) Knowledge engineering interface; and
- 4) Subject-matter expert interface.

We now begin by discussing items 1 and 2—the backends.

3.1 Query Control and the CITKA Backends

The interaction between the SME and CITKA is initiated by questions that address each of the six elements composing the context base. It is highly intuitive for an SME to provide much of this information in a query session. CITKA progresses through each of those required items of knowledge until all relevant knowledge has been acquired. Of course, one cannot expect most SMEs to express action functions directly in a computing language. This is typically done later by the KE. However, many of these functions will be standard (e.g., distance between two points), making them easily reusable across different applications. This will reduce the burden on the KE.

CITKA uses rules to compose the appropriate queries. The *query rule base backend* is the rule-based system used to guide the intelligent dialog with the SME. The *subject-matter expert interface* module maps into the query rule base backend. Its rule base structure

is designed to reflect the CxBR hierarchy, i.e., the Mission Context, the Major Contexts, the Sub-contexts, etc. The CxBR structure naturally divides CITKA's rules into modules that reflect these activities/decisions and proceeds to query the expert in a top-down fashion. A new query module is not started until the previous one has been completed. This allows the interaction with the expert to progress from the general to the specific. This top-down interaction can make the expert feel at ease, and he can naturally acquire the mindset and framework of the problem. Furthermore, details that may not have been clear to him earlier in the session may become clear later but in the meantime do not impede the expert from describing the knowledge.

To start, the expert is prompted for a general outline of the mission context and its objective. This outline will serve the purpose of a skeleton that is fleshed out by progressively adding information that is more specific. The interaction with the expert can be rather lengthy; therefore, CITKA allows the session to be interrupted and thereafter seamlessly resumed in subsequent sessions. The questions deal with the following knowledge items, in the following order.

- 1) Mission description (Mission Context—item 1 of the six items in section 2.1)
- 2) Other entities or objects involved in the scenario (item 5 in section 2.1)
- 3) Major Contexts (item 2 in section 2.1)
- 4) Actions associated with each Major Context (item 3 in section 2.1)
- 5) Transitions to the next Major Context (item 4 in section 2.1)
- 6) Sub-contexts involved with each Major Context (item 2 in section 2.1)
- 7) Actions associated with each Sub-context (item 3 in section 2.1)
- 8) Transitions to other Sub-contexts and the associated Major Contexts (item 4 in section 2.1)
- 9) Short-/long-term situational memory of relevant events/conditions
- 10) Helping functions needed to support/define abstract concepts used for Context/Sub-context actions, Context/Sub-context transition decisions or situational memory (item 6 in section 2.1)

Some aspects of a context base are easier to capture and convert directly into useful code than are others. More specifically, those responses that are in the form of unconstrained text will likely require interpretation by a KE for conversion into useful code. At some future time, a natural language processing module could be integrated to automate this process, even if only partially. However, for the time being, it relies on human effort. The system continues asking questions

of the SME until the knowledge for all six items in the list above have been completely accounted for. At this point, the system has done all it can and now turns over completion of the context base to a KE. The KE can use the knowledge engineering interface to complete the work.

The knowledge engineering database backend holds the evolving context base as it gradually becomes developed by CITKA with the knowledge supplied by the SME and the KE. The knowledge engineering interface module maps into the knowledge engineering database backend module. The requirements for these subsystems are quite different and are treated separately in the following description.

The knowledge engineering database backend is central to the system because it is the repository of the context base for the agent. It is written in COOL, the object-oriented extension to the CLIPS tool. The database defines a class for each entity modeled in the CxBR CGF. These entities are:

- The Mission Context (item 1 in section 2.1);
- The Major Contexts associated with the Mission Context (item 2);
- The Sub-contexts of the Mission Contexts (item 2);
- The transition criteria that state when agent control should transition from one Major Context to another—these are generally implemented as rules (sentinel rules) (item 4);
- The action definitions that indicate what action to take within a context (item 3);
- Memory variables used by the CGF agent;
- Helping functions that perform low-level, basic actions used by higher-level functions, or that return state information for the CGF agent such as location, speed, and heading (item 5); and
- The entity objects used by the CGF agent (item 6).

The following guidelines are designed to maintain conciseness in the context base being developed, and the SMEs should be aware of them:

- 1) All Major Contexts should be associated with the Mission.
- 2) Transition criteria and action definitions can be shared across contexts.
- 3) Deleting a Major Context eliminates all of its associated Sub-contexts, but only if they are not associated with any other Major Context.
- 4) Deleting a Major Context or Sub-context will eliminate its action definitions and transition criteria only if they are not used by any other context.

The persistent property of the objects is provided

by the save-instances and load-instances CLIPS commands without need for any modification. This has the advantage of saving to text files that can be easily ported from implementation to implementation.

During the development of the query rule base, it was found that query rule activation was good for developing complex interfaces that adapt themselves to the SME needs. However, rules were overkill and were cumbersome to use when the sequence of screens to implement was simple and linear. To avoid this unnecessary complexity, a scripting capability was added on top of the query rule base, effectively dividing the query rule base into disjoint sets that are executed sequentially. This allows CITKA to provide, on one extreme, a linear interface where each rule base set activates only one rule. Alternatively, on the other extreme, CITKA can invoke a complex rule base that interacts intelligently with the SME to extract knowledge from her. The scripting capability was dubbed the “virtual machine” because it implements a small programming language within the CLIPS application itself.

A clearer picture of this internal scripting language can be seen in the following code snippet.

```
(begin
  weather
  location
  constraints
  objective-achieved
  mission-check-review
  (cond rule-continue-1
    (case "review"
      mission-review
    )
  )
)

objects-get-friendly
objects-get-enemy
objects-get-neutral
objects-get-other
objects-review1
(forall objects
  objects-get-description
  objects-get-attributes-descr
  (forall object-attributes
    objects-get-details
  )
)
...
)
```

This is the starting sequence of the SME interface that defines the program to execute, where each entry corresponds to an independent rule set. For example, “objects-get-friendly” is the set of query rules that ask the user to describe the friendly entities that are to be part of the environment.

3.2 User Interface

CITKA provides an interface for SME interaction. It is through this interface that the queries are presented to the SME and his/her responses are captured. CITKA also provides an interface for a KE to complete and/or refine the knowledge entered by the SME. However, no queries are ever presented to the KE. The opening screen gets the process started. It first asks whether the user is an SME or a KE. Figure 2 depicts the basic opening screen. The user merely points to the button marked either subject-matter expert or knowledge engineer and clicks on it to set the appropriate context for the session. This active *user mode* will be visually highlighted for the duration of the session, or until the user selects the other one.

All subsequent CITKA screens consist of three areas; see Figure 3. The first area is the *control area*, and is located vertically on the left-hand side of the screen. This area, called the *current area of interest*, contains the active buttons that can move the user between the different parts of the knowledge base being created. The active button will always be visually highlighted.

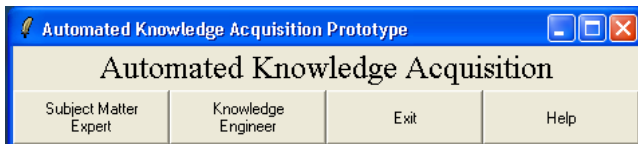


Figure 2. Opening screen

In addition, just below the control area, the same set of buttons shown in Figure 2 appears, indicating the user mode of the system. Since the more serious research issues involve providing assistance to the SME via a sequence of intelligent questions (by far the more interesting of the two user modes), from this point on we will assume we are in the SME mode unless otherwise indicated.

The second area is the *query area*. It covers the upper half of the remaining screen area not covered by the control area. The questions and instructions to the user are displayed here. The area is scrollable, with a scroll bar at the right-hand side. It contains text for the user to read—it is not active for user input in any way. The queries are in natural language.

The third area is the *response area*, where the SME will respond to the system's queries. It covers the lower half of the remaining screen and is just below the query area. Like the query area, the response area also scrolls, with a scroll bar at the right-hand side. Unlike the query area, however, the response area is active for user input. Depending on the response required from the SME, the response area will provide space for an unconstrained textual response, or it will specifically seek indication from the user in terms of multiple choice, true or false, or simple one-word answers. After the user's response is entered, the query and response areas will change to reflect the next query.

The responses from the SME can be in three forms: 1) unstructured text, 2) structured text (simple one- or two-word answers), or 3) multiple choice. The

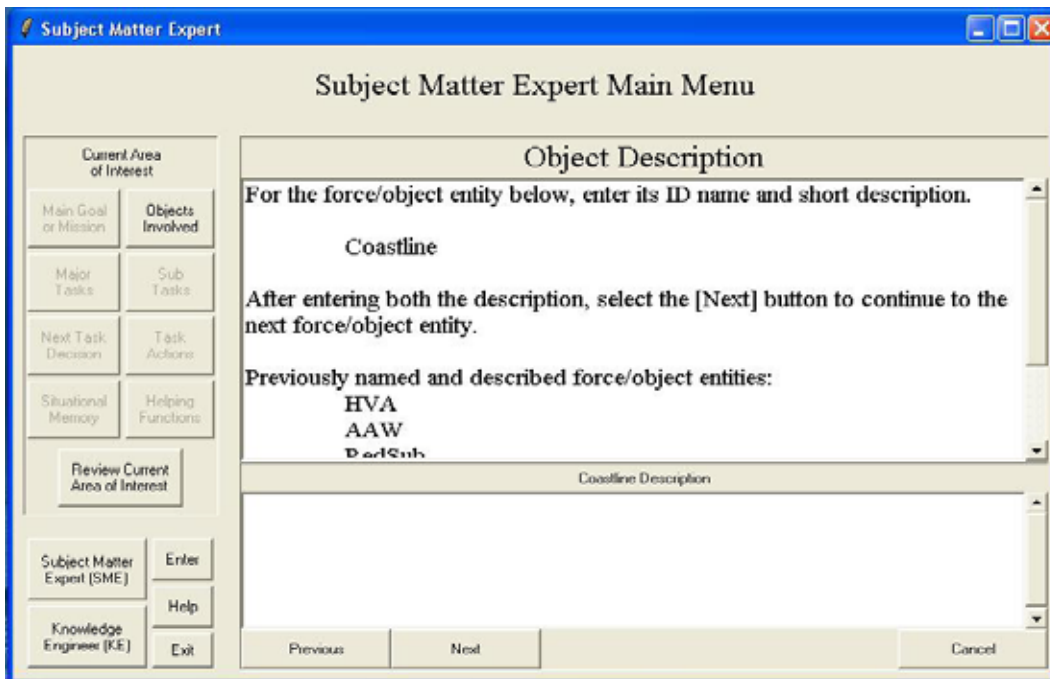


Figure 3. CITKA screen

unstructured text is used for describing algorithms or processes used by expert performers. This form of input is not processed further by CITKA and is either passed on to the KE for action or simply used as documentation for certain actions. The structured text represents attributes, values of attributes, names of contexts, objects, or functions. These are readily used by CITKA to compose classes and/or rules. Multiple choice questions are those that present the SME with several choices, and he must select one or more of these. These are also employed by CITKA to compose code.

The SME interface is a GUI for the query rule base backend. Unfortunately, our implementation for the query rule base (CLIPS) is not naturally GUI oriented, so a flexible interface that could access the CLIPS system in a more effective manner had to be developed. This interface was coded in Tcl/Tk, a scripting language with strong user interface and program integration capabilities. It has the advantage of being cross platform (x86 Windows, Mac, UNIX XWin) source-compatible. It was originally developed by Sun Microsystems and is now maintained by Scriptics Corp. The interface for the KE is merely a screen editor and is not described further. Other implementation options exist besides the CLIPS/Tcl/Tk. Specifically, the Java Expert System Shell (JESS), inspired by and very similar to CLIPS, could be used alternatively. Its Java base provides the cross-platform capability of Tcl/Tk.

In summary, the CITKA prototype has been designed and built to naturally integrate with the CxBR structure. It allows a great deal of flexibility in its internal design. This flexibility is necessary because the subject-matter expert interface must dynamically produce interfaces that correspond to the questions and rules fired in the query rule base. Nevertheless, it should be noted that while CITKA is intelligent in the questions it asks the SME, it does not contain any advances in automated knowledge acquisition. Its advantage lies in its use of the highly structured CxBR modeling paradigm that serves as its basis. This permits the elicitation process to proceed in a very intuitive fashion.

4. Evaluation of the CITKA System

To evaluate the effectiveness of our approach, we subjected the CITKA prototype to a rigorous evaluation. The objective was to determine the reduction in human effort (person-hours) over what would be required if manual methods were used. By *manual methods*, we mean interviews between the KE and the SME. Our evaluation consisted of theoretically estimating the savings in human effort and then verifying these estimates by exercising the prototype system. There are very few instances of published data on test

results in this subject [16, 17], and even fewer that deal with tactical knowledge. Noy, Grosso, and Musen [44] expressed the general difficulty in evaluating knowledge acquisition tools as a result of the difficulty in procuring a group of SMEs who are willing and able to participate in experiments that may last several days. In fact, our work suffered from the same difficulty, forcing us to make educated estimates and validate them through extrapolation of limited experiments. Therefore, opportunities to compare our results with relevant published data were not available.

Two main questions were addressed: 1) How much KE person-hour effort to develop a context-based agent for a particular tactical mission can be eliminated using CITKA? 2) What effect does the use of CITKA have on the SME person-hour effort for the same tactical mission? It would have been desirable to evaluate the quality of the knowledge elicited through CITKA and compare it with that obtained through traditional means. However, because of the same difficulties discussed above in procuring SME participation, we were unable to complete the model being developed through CITKA, and therefore had no basis for a rigorous comparison. We leave this important aspect of automated knowledge acquisition for future research.

The baseline for comparison was the agent-building effort involved in the Human Behavior Representation Challenge project, sponsored by the Defense Modeling and Simulation Office (DMSO) in 2001. This project entailed the development of a submarine agent involved in a surface asset protection mission. A vignette was defined by DMSO to include the mission objectives and constraints. The resulting CxBR-based agent operated on a standard test bed for an apples-to-apples comparison with other modeling approaches. This knowledge acquisition effort was done in the traditional way—interviewing SMEs and reviewing published materials.

The vignette involves a developing situation with an unfriendly coastal nation with whom tensions have progressively risen. A naval task force (blue) is dispatched to patrol the coast of this country (red) as a show of force. It can also serve as an attacking force if hostilities break out. The naval task force is composed of an aircraft carrier (the high value asset, or HVA), two anti-aircraft picket ships to protect the carrier from airborne threats, and a submarine to protect the task force from naval threats—specifically, enemy diesel submarines. The blue submarine is tasked with entering enemy waters to monitor activity in an enemy port located on the coast being patrolled. The submarine is also to protect the surface blue vessels if they are threatened. The rules of engagement are to not fire at the enemy unless: 1) the enemy fires first, 2) a red submarine

comes within firing range of surface assets, or 3) a red submarine performs two threatening maneuvers in sequence. The red force consists of up to three diesel submarines, whose intentions may be unclear. Scripts developed by DMSO controlled the red submarines, causing the blue submarine agent to react. In addition, the blue submarine agent was ordered to report on enemy activity at the port every 24 hours.

We now compare CITKA to the traditional means of knowledge acquisition. Firstly, in simple terms, this analysis compares the time actually taken by a KE to develop the full agent model of the blue submarine through the traditional manual process to an estimate of the time taken to develop the same agent by having an SME interact with CITKA. This first analysis focused solely on the KE's effort. Secondly, we sought to determine the effect of CITKA on the level of effort expended by the SME. That is, does the use of CITKA impose new burdens on the SME, thereby increasing his workload? We explore these two issues next, but first we describe the benchmark data used for comparison.

4.1 Benchmark Case: Traditional Knowledge Acquisition

The first set of data recorded was the benchmark data. That is, the actual time (in person-hours) spent by a KE in manually acquiring and codifying the knowledge of the blue submarine agent in the aforementioned mission. These data are shown in Table 1 below.

Table 1. Actual KE Hours Spent on Agent Development

Task	Hours	Task Description
1	157.0	Learning curve on CxBR and framework by KE
2	134.5	Acquisition of domain knowledge by KE
3	243.3	Agent design, including definition of algorithms by KE
4	150.0	Agent development—coding of agent by KE

The resulting total for all four tasks was approximately 685 person-hours. We refer to this value as the ACTUAL-SPENT-EFFORT. Therefore,

$$\text{ACTUAL-SPENT-EFFORT} = 685.$$

However, not all this effort can be reduced through CITKA. The next section explores what can and cannot be reduced.

4.2 Reduction in KE Effort

To determine the how much KE effort could be saved through the application of CITKA, we examine the four tasks that comprise the ACTUAL-SPENT-EFFORT. We then estimate how many of these hours would still be necessary if the SME used CITKA instead. This is admittedly a hypothetical estimate, but we will use the most conservative estimate whenever possible. As a result of this analysis, we estimate another variable, the REVISED-SPENT-EFFORT-WITH-CITKA. This value will reflect the hours of human KE effort *that would still be required* of the KE when CITKA is used by the SME. The lower this number, the more effective CITKA is.

Task #1: Whether to include the hours spent on Task #1 in the REVISED-SPENT-EFFORT-WITH-CITKA presents an interesting argument. Even with the use of CITKA, a KE still needs to understand how CxBR and its associated framework operate in order to refine and complete the knowledge. On the other hand, once a KE becomes familiar with CxBR and the framework, this effort would no longer be required in the development of future agents. In keeping with our stated conservatism in estimates, we deemed that this effort was not avoidable through CITKA, and thus *included* it in toto in the REVISED-SPENT-EFFORT-WITH-CITKA estimate. Therefore, initially, we set the REVISED-SPENT-EFFORT-WITH-CITKA = 157 hours.

Task #2 is the process of acquiring knowledge from the expert or other sources. This includes defining the mission and enumerating the required Major and Sub-contexts. It also includes enumerating and describing the objects involved in the mission (e.g., red submarines, aircraft carrier). If an SME were to use CITKA, the majority of this effort would not be necessary on the part of the KE, as CITKA would be doing that as part of its interaction with the SME. Therefore, it is logical to *not include* the hours from Task #2 in the REVISED-SPENT-EFFORT-WITH-CITKA value. However, it is also reasonable to expect that the KE will need to spend some time refining the context base, and in doing so will need to obtain clarifications from the SME. We will, therefore, include 20% of these hours in our calculation of REVISED-SPENT-EFFORT-WITH-CITKA. That amounts to approximately 27 hours. Therefore, we now set the REVISED-SPENT-EFFORT-WITH-CITKA = 157 + 27 = 184 hours.

Task #3 represents building the objects and defining the algorithms, functions, and rules that compose the agent. This would also be directly replaced by CITKA, so it clearly should not be included in the REVISED-SPENT-EFFORT-WITH-CITKA value. The only question about Task #3 is whether the time taken to compose the algorithms should be included in that total. This question arises because it is highly unlikely that a typical military SME

is capable of composing an algorithm. The number of hours specifically spent in algorithm development was not tracked in our benchmark case, making this estimate difficult. Therefore, in keeping with our conservative approach, we will include 20% of these hours in our *REVISED-SPENT-EFFORT-WITH-CITKA* estimate to account for algorithm development by the KE. That comes to approximately 49 hours. As will be seen later, our SME was in fact able to compose algorithms via CITKA, making this question irrelevant vis-à-vis our later comparison. Updating the *REVISED-SPENT-EFFORT-WITH-CITKA*, we now set it to $157 + 27 + 49 = 233$ hours.

Task #4: In the great majority of situations, the actual coding of the model cannot be done by the SME and is therefore the responsibility of the KE. However, much of the context-based model merely requires creation of classes with the attributes being assigned values. This can easily be done by CITKA, resulting in compiled source code. With somewhat less confidence, we believe that the sentinel rules that incorporate the transition criteria to control the context transition process can also be built automatically from the transition criteria. In fact, Henninger [32] was able to automatically write CLIPS rules with her prototype, giving us confidence about this assertion. The area of greatest difficulty would be coding action functions. We therefore arguably assume that 70% of the code could be automatically generated. Further assuming a linear relation between lines of code and time, the use of CITKA would replace 70% of the 150 hours taken by the benchmark numbers. This would require 30% of the effort to still be done by the KE. This would amount to nearly 45 hours of coding effort expended for the benchmark mission by the KE. Therefore, the final value of *REVISED-SPENT-EFFORT-WITH-CITKA* = $157 + 27 + 49 + 45 = 278$ hours of KE effort required in the presence of CITKA. The final value can now be calculated as:

REVISED-SPENT-EFFORT-WITH-CITKA = 278 hours of KE effort.

Comparing the *REVISED-SPENT-EFFORT-WITH-CITKA* number to the *ACTUAL-SPENT-EFFORT* by the KE, the effort of the KE in light of the use of CITKA is now approximately 40% of the original effort. A reduction of almost 60% could conservatively be attributed to the use of CITKA.

4.3 Reduction of SME Effort

This part of the estimate seeks to measure the effect of using CITKA on the SME. Consequently, the second set of data recorded was an estimate of how long it would take for an SME comfortable with the use of a computer to respond to the questions posed by CITKA in its ultimate form. Note that the prototype was not used in this analysis. Once again, the benchmark for comparison was the naval mission described above.

These estimates are based upon careful analysis of the SME's tasks and include a low and a high estimate. These respectively assumed 15 and 30 seconds as the low and high estimates for entry of each name, value, etc.; 10 seconds for each checkbox checked; and 2 and 4 (low and high) minutes per line of text entered. The text entry assumed 80 characters per line and a typing speed of 20 and 40 characters per minute (low and high). This estimate includes time for typing and scrolling, as well as thinking. Table 2 shows these estimates.

The *cumulative* columns show the running total in person-hours for the progressing effort, with and without entering the algorithms. The idea behind algorithm description is that if the SME is capable of developing an algorithm and inputting it into CITKA, then the rightmost column reflects the running total. However, it is unrealistic to believe that the typical military SME will be able to express an algorithm in pseudo-code, readily transferable to a computer program. More likely, he will describe it in running text, and the KE would have to generate an algorithm from his description. In this case, the middle column is the right one to use in the comparison. However, we consider the rightmost column to be the more conservative estimate of the two, vis-à-vis the SME, and use it in our analysis.

The results shown in the rightmost column of Table 2 indicate a total of nearly 71 hours, assuming an SME who responds quickly, and 136 hours for a slow one. These values assume a capable SME taking the time to develop algorithms—the worst case for an SME in terms of hours expended. A fair comparison of this number would be the documented values of Task #2 in Table 1, as this is the only task of these four that would involve an SME. If a one-on-one interaction between an SME and KE is assumed, then the SME would have spent 134.5 hours (Table 1, Task 2). This expected effort of 134.5 hours by an SME in Task #2 compares quite well with the estimate of 136.4 hours for a particularly slow SME who also spends the time to develop algorithms. The optimistic value of 70.7 hours, which also includes algorithm development, reduces the SME's effort to nearly 53% of the actual values (70.7/134.5). In summary, the conservative conclusion can be reached that the effort expended by an SME is roughly comparable, and potentially much lower under some circumstances.

4.4 Prototype Evaluation

However, these are only estimates, albeit carefully considered ones. An empirical evaluation was done to verify these estimates so as to validate the conclusions made here—that the use of CITKA greatly reduces the

Table 2. Summary of estimates for SME inputs (in person-hours)

Task Description	Individual Task (Low – High)			Cumulative (w/o Algorithms) (Low – High)			Cumulative (with Algorithms) (Low – High)		
Overall Mission Description (no algorithms necessary)	3.2	–	6.3	3.2	–	6.3	3.2	–	6.3
Forces/Object Definition (no algorithms necessary)	6.0	–	11.8	9.2	–	18.1	9.2	–	18.1
Major Context Descriptions									
w/o Algorithm Definition	7.3	–	14.4	16.5	–	32.5			
Algorithm Definition	3.0	–	5.9						
Total for MC Descriptions	10.3	–	20.3				19.5	–	38.5
Sub-context Descriptions									
w/o Algorithm Definition	16.0	–	31.6	32.5	–	64.1			
Algorithm Definition	6.6	–	13.2						
Total for SC Descriptions	22.57	–	44.79				42.0	–	83.3
Situational Memory	1.3	–	1.8	33.8	–	65.9	43.4	–	85.0
Helping Functions Descriptions									
w/o Algorithm Definition	11.9	–	20.4	45.7	–	86.3			
Algorithm Definition	15.5	–	31.0						
Total for HF Descriptions	27.4	–	51.4				70.7	–	136.4
Overall Totals				45.7	–	86.3	70.7	–	136.4

Table 3. Results for Major Context descriptions

Screen	Total Time	Number of Entries	Average Time on Screen
choose-scenario	0:00:54	11	
Description	0:00:06	1	
major-action-intro	0:04:34	7	0:00:46
major-task-get-action-algorithm	1:52:55	48	0:18:49
major-task-get-action-description	0:27:43	37	0:04:37
major-task-get-action-input-description	1:22:57	317	0:02:08
major-task-get-action-inputs	0:59:04	36	0:09:51
major-task-get-universal-names	0:00:12	3	0:00:04
major-task-get-validity-algorithm	0:44:32	18	0:02:28
major-task-get-validity-input-description	0:32:27	163	0:00:12
major-task-get-validity-inputs	0:24:00	33	0:00:44
major-task-review	0:05:19	11	0:00:29
reactive-major-task-get-description	0:00:05	5	0:00:01
universal-major-task-get-description	0:07:38	18	0:00:25
Total	6:42:26	708	

human effort required to build the models of human behavior in tactical situations. An SME was asked to enter his knowledge on the same sea vignette mission into the CITKA prototype, and his efforts were carefully monitored and quantified. The purpose of this exercise was to verify the estimates shown in Table 2.

The description of Table 3 and its fields is as follows:

- *Screen* indicates the particular input screen that was used in CITKA by the SME. The name is arbitrary but reflects to some extent the context of the input being acquired.
- *Total time* indicates how much time was spent by the SME on that particular screen, in hours:minutes:seconds.
- *Number of entries* indicates how many times the SME entered that particular screen. The amount of entries reflects the navigation of the expert necessary to review information on other screens and is higher than the amount of logical inputs being made. For example, the amount of actions for Major Tasks was six, but the screen for capturing major task actions was entered a total of 37 times.
- *Average time on screen* indicates the average time spent on each screen, given the amount of times it was accessed.

One problem faced when evaluating the prototype is that testing can only be done in real time. That is, we must measure the actual time spent by an SME interacting with CITKA to create the context base. In our case, it is the context base used for the aforementioned sea scenario. Given that the estimate for the time needed to develop this context base is between 71 and 136 hours, we estimate that the testing time necessary to completely evaluate this prototype would be on the order of 100 ± 30 hours. This represents between nearly 2 and 3.25 person-weeks of effort for the SME alone. Our SME, who agreed to assist in this matter on a pro bono basis, could only dedicate one to two hours per week to this effort. That would require a period of between one and one-and-a-half years to collect the complete data—a clearly unrealistic option. Faced with no other alternative, we chose instead to evaluate a small part of the scenario, and then apply extrapolation. That would provide us with a strong sense of whether our estimates were realistic or not. Table 3 depicts the actual times taken by the SME using the CITKA prototype in developing the definition of one Major context.

To measure the actual clock time the SME spent in the various CITKA screens, a timing program was installed and run in the background. This program

monitors the mouse clicks and keeps track of the time the SME took on each particular task. That eliminated the need for manual time keeping by the research staff, a meticulous and less accurate process. The actual testing only encompassed the *overall mission description* (Mission Context dialog) and *Major Context descriptions* (Major Context dialog). Results for Major Context descriptions are given in Table 3.

We are particularly interested here in validating the time estimates for SME involvement made in section 4.3. As can be seen, the total time of SME involvement for this part of the scenario is 6 hours 42 minutes and 26 seconds. This included developing the required algorithms (which our SME was able to do). This is a bit less than the estimated best-case time of 7.3 hours to do the same thing *without* algorithm definitions, and significantly less than the estimated time *with* algorithm definitions (10.3 hours). This implies that there was some overestimation in our estimated times for the SME to use CITKA to input the screens. In any case, the results tend to confirm that the estimates for using CITKA are quite credible, if a bit pessimistic, giving validity to our hypothesis that there is no added burden on the SME by using CITKA. In fact, our estimates may have been overly pessimistic, suggesting a decrease in SME effort through CITKA.

It should be noted that the SME used was also the same individual who specified (but not developed) CITKA. While this may have given him a distinct advantage over someone unfamiliar with CITKA, it is our opinion that the numbers provide acceptable results in spite of this. This is justified by the fact that even a 50% increase in the time taken by the SME would have placed the results just above the lower end of the range of estimates (10.1 versus 10.3 hours) if the algorithms were included in the estimate.

As mentioned earlier, our experiment suffered from a difficulty common to most if not all such experiments in automated knowledge acquisition. SMEs are hard to find in sufficient numbers to make a truly statistically meaningful conclusion. Moreover, those that can be found are typically unavailable for a sufficient length of time to carry out a non-trivial experiment. Noy, Grosso, and Musen [44] express this difficulty candidly. Nevertheless, engineering analysis traditionally permits evaluation of concepts (designs, operation) in ways other than empirical testing in cases when the latter is impossible. This is what we have done. We believe that our evaluation, while short of being empirically complete, is appropriate and provides a realistic evaluation of the CITKA concept.

5. Summary and Conclusions

The results indicate that model development using CITKA significantly facilitates the knowledge acquisition task for building tactical behavior agents in simulations. In this paper, we described the underlying concepts for building the CITKA system to do this very thing. Furthermore, we described a prototype system used to prove CITKA's technical feasibility and evaluate the potential effectiveness of the system. The results indicate significant savings in effort spent in developing agents that exhibit human tactical behavior. Our conservative estimates indicate a reduction of 60% of the time and effort spent by the KE. Moreover, we estimate that there will be no negative impact on the involvement of the SME. Limited testing with the actual prototype indicates that these estimates are indeed realistic, if not a bit pessimistic.

The commercial development of the CITKA prototype would facilitate development of CGFs by making them easier to build. While the current prototype was designed and developed in accordance with current software engineering practices, it is not considered "commercial-grade" and would require some re-evaluation and possibly some rewriting to make it so. This is currently in the plans for our future work. CITKA does, however, commit the user to the CxBR modeling paradigm being employed as the basis of the resulting model.

CxBR has never been rigorously compared head-to-head with other modeling paradigms in terms of model effectiveness. Brown came the closest to doing so in his thesis [34], but it was not a rigorous comparison. Such a comparison is beyond the scope of this paper and is left for future research. Nevertheless, based on our experiences as well as on Brown's results, it is our opinion that CxBR offers at least an equal, if not superior, means of modeling human performance. Of course, it must be noted that model effectiveness often rests more with the modeler than with the modeling paradigm. The latter can only facilitate the construction of the model and provide a rich environment for easily representing the various aspects of human behavior. At this we feel that CxBR is quite effective.

Other future research could involve a study of how typing the knowledge into the computer directly by the SMEs differs, in terms of acquisition effectiveness, from stating it orally to a KE.

6. References

- [1] Anderson, J. R., M. Matessa, and C. Lebiere. "ACT-R: A Theory of Higher Level Cognition and its Relation to Visual Attention." *Human Computer Interaction* 12, no. 4 (1997): 439–462.
- [2] Zubritsky, M. C., W. W. Zachary, and J. M. Ryder. "Constructing and Applying Cognitive Models to Mission Management Problems in Air Anti-Submarine Warfare." In *Proceedings of the Human Factors Society 33rd Annual Meeting*, 129–133. 1989.
- [3] Gonzalez, A. J. and R. Ahlers. "Context-Based Representation of Intelligent Behavior in Training Simulations." *Transactions of the Society of Computer Simulation* 15, no. 4 (1998): 153–166.
- [4] Laird, J. E., A. Newell, and P. S. Rosenbloom. "Soar: An Architecture for General Intelligence." *Artificial Intelligence* 33, no. 1 (1987): 1–64.
- [5] Tambe, M., W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, and P. S. Rosenbloom. "Intelligent Agents for Interactive Simulation Environments." *AI Magazine* (Spring 1995): 15–39.
- [6] Shaw, M. L. G. "PLANET: Some Experience in Creating an Integrated System for Repertory Grid Application in a Microcomputer." *Int. J. of Man-Machine Studies* 17 (1982): 345–360.
- [7] Boose, J. H. "Personal Construct Theory and the Transfer of Human Expertise." In *Proceedings of the National Conference on Artificial Intelligence*, 27–33, AAAI-84, 1984.
- [8] Boose, J. H., and J. Bradshaw. "Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge-Acquisition Workbench for Knowledge-Based Systems." *Int. J. Hum.-Comput. Stud.* 51, no 2 (1999): 453–478.
- [9] Parsaye, K. "Acquiring and Verifying Knowledge Automatically." *AI Expert* (May 1988): 48–63.
- [10] Kahn, G., S. Nolan, and J. McDermott. "MORE: An Intelligent Knowledge Acquisition Tool." In *Proceedings of the 1985 International Joint Conference on Artificial Intelligence*, IJCAI-85, Los Angeles, CA, 1985.
- [11] Marcus, S., J. McDermott, and T. Wang. "Knowledge Acquisition for Constructive Systems." In *Proceedings of the 1985 International Joint Conference on Artificial Intelligence*, IJCAI-85, Los Angeles, CA, 1985.
- [12] Farquhar, A., R. Fikes, and J. Rice. "Tools for Assembling Modular Ontologies in Ontolingua." In *Proceedings of the 1997 National Conference on Artificial Intelligence*, 436–441, AAAI-97, 1997.
- [13] Eriksson, H., Y. Shahar, S. W. Tu, A. R. Puerta, and M. A. Musen. "Task Modeling with Reusable Problem-Solving Methods." *Artificial Intelligence* 79 (1985): 293–326.
- [14] Canas, A., D. B. Leake, and D. C. Wilson. "Managing, Mapping and Manipulating Conceptual Knowledge." AAAI Workshop Technical Report WS-99-10: Exploring the Synergies of Knowledge Management & Case-Based Reasoning. Menlo, CA: AAAI Press, 1999.
- [15] Bareiss, R., B. W. Porter, and K. S. Murray. "Supporting Start-to-Finish Development of Knowledge Bases." *Machine Learning* 4 (1989): 259–283.
- [16] Yost, G. R. "Acquiring Knowledge in Soar." *IEEE Expert* 8, no. 3 (1993): 26–34.
- [17] Kim, J., and Y. Gil. "User Studies of an Interdependency-Based Interface for Acquiring Problem-Solving Knowledge." In *Proceedings of the International Conference on Intelligent User Interfaces*, IUI-2000, January 2000.
- [18] Shutte, P. C. "Definitions of Tactical and Strategic: An Informal Study." NASA Technical Report, NASA/TM-2004-213024, Nov. 2004, <<http://techreports.larc.nasa.gov/ltrs/PDF/2004/tm/NASA-2004-tm213024.pdf>>.
- [19] Thorndike, P. W., and K. T. Wescourt. "Modeling Time-Stressed Situation Assessment and Planning for Intelligent Opponent Simulation." Final Technical Report PPAFTR-1124-84-1, Sponsored by the Office of Naval Research, July 1984.
- [20] Endsley, M. "Towards a Theory of Situational Awareness in Dynamic Systems." *Human Factors* 37, no. 1 (1995): 32–64.

- [21] Endsley, M. R. "Design and Evaluation for Situation Awareness Enhancement." In *Proceedings of the Human Factors Society 32nd Annual Meeting*, 97–101, Human Factors Society, Santa Monica, CA, 1988.
- [22] Randolph, W. "An Instrumented Knowledge Acquisition Process for Modeling and Simulation Development." In *Proceedings of the Simulation Interoperability Workshop*, Orlando, FL, Spring 2002.
- [23] Delugach, H. S., and D. J. Skipper. "Knowledge Techniques for Advanced Conceptual Modeling." In *Proceedings of the Ninth Conference on Computer Generated Forces and Behavior Representation*, Orlando, FL, May 2000.
- [24] Gonzalez, A. J., M. Georgiopoulos, R. F. DeMara, A. E. Henninger, and W. Gerber. "Automating the CGF Model Development and Refinement Process by Observing Expert Behavior in a Simulation." In *Proceedings of the 7th Conference on Computer Generated Forces and Behavior Representation*, Orlando, FL, July 1998.
- [25] Van Lent, M., and J. Laird. "Learning by Observation in a Tactical Air Combat Domain." In *Proceedings of the 8th Conference on Computer Generated Forces and Behavior Representation*, Orlando, FL, May 1998.
- [26] Morrison, J. D. "Real Time Learning of Doctrine and Tactics Using Neural Networks and Combat Simulations." *Military Operations Research* 2, no. 3 (1996): 45–60.
- [27] Hovland, G. E., P. Sikka, and B. J. MacCarragher. "Skill Acquisition from Human Demonstration Using a Hidden Markov Model." In *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, 2706–2711, 1997.
- [28] Sidani, T. A., and Gonzalez, A. J. "A Framework for Learning Implicit Expert Knowledge Through Observation." *Transactions of the SCS* 17, no. 2 (June 2000): 54–72.
- [29] Sammut, C., S. Hurst, D. Kedzier, and D. Michie. "Learning to Fly." In *Proceedings of the Ninth International Conference on Machine Learning*, 335–339, Aberdeen, 1992.
- [30] Henninger, A. E. "Neural Network-Based Movement Models to Improve the Predictive Utility of Entity State Synchronization Methods for Distributed Simulations." Ph.D. diss., May 2001.
- [31] Pomerlau, D., C. Thorpe, D. Longer, J. K. Rosenblatt, and R. Sukthankar. "AVCS Research at Carnegie Mellon University." In *Proceedings of Intelligent Vehicle Highway Systems America 1994 Annual Meeting*, 257–262, 1994.
- [32] Henninger, A. E., and A. J. Gonzalez. "Automated Acquisition Tool for Tactical Knowledge." In *Proceedings of the 10th Annual Florida Artificial Intelligence Research Symposium*, 307–311, Daytona Beach, FL, May 1997.
- [33] Grejs, P. F. "Autonomous Automobile Behavior Through a Context-Based Approach." Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, 1998.
- [34] Brown, J. C. "Application and Evaluation of the Context-Based Reasoning Paradigm." Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, July 1994.
- [35] Gumus, I., and A. J. Gonzalez. "A Threat Ranking Algorithm for Multiple Intelligent Entities in a Simulated Environment." In *Proceedings of the Florida Artificial Intelligence Research Society Conference*, Orlando, FL, 1999.
- [36] Proenza, R. "A Framework for Multiple Agents and Memory Recall within the Context-Based Reasoning Paradigm." Master's Thesis, ECE Dept., University of Central Florida, Orlando, FL, 1997.
- [37] Norlander, L. "A Framework for Efficient Implementation of Context-Based Reasoning in Intelligent Simulation." Master's Thesis, Electrical and Computer Engineering Dept., University of Central Florida, Orlando, FL, 1999.
- [38] Devero, L., and A. J. Gonzalez. "Collaborative Teamwork Representation in Context-Based Reasoning." In *Proceedings of the Conference on Computer Generated Forces and Behavior Representation*, Norfolk, VA, 2001.
- [39] Gallagher, A., and A. J. Gonzalez. "Modeling Platform Behaviors Under Degraded States." In *Proceedings of the Inter-service/Industry Training Systems and Education Conference (I/ITSEC)*, Orlando, FL, December 2000.
- [40] Turner, R. M. "Context-Sensitive Reasoning for Autonomous Agents and Cooperative Distributed Problem Solving." In *Proceedings of the 1993 IJCAI Workshop on Using Knowledge in Its Context*, Chambéry, France, 1993.
- [41] Turner, R. M. *A Model of Explicit Context Representation and Use for Intelligent Agents*. Springer-Verlag, 1999.
- [42] Brézillon P., J.-Ch. Pomerol, and I. Saker. "Contextual and Contextualized Knowledge: An Application in Subway Control." *Int. J. Hum.-Comput. Stud.* Special Issue on Using Context in Applications, 48, no. 3 (1998): 357–373.
- [43] Bass, E. J., J. P. Zenyuh, R. L. Small, and S. T. Fortin. "A Context-Based Approach to Training Situation Awareness." In *Proceedings of the Third Annual Symposium on Human Interaction with Complex Systems*, 89–95. Los Alamitos, CA: IEEE Computer Society Press, 1996.
- [44] Noy, N. F., W. Grosso, and M. A. Musen. "Knowledge-Acquisition Interfaces for Domain Experts: An Empirical Evaluation of Protégé-2000." In *Proceedings of the SEKE-2000 Conference*, 2000.

Author Biographies

Dr. Avelino Gonzalez received his Bachelor's and Master's degrees in Electrical Engineering from the University of Miami, in 1973 and 1974, respectively. He obtained his Ph.D. from the University of Pittsburgh in 1979 also in Electrical Engineering. He is a professor in the School of Electrical Engineering and Computer Science at the University of Central Florida, specializing in artificial intelligence and simulation. He is the co-director of the Intelligent Systems Laboratory at the same institution.

Dr. William Gerber obtained his Ph.D. in Computer Engineering at the University of Central Florida in 2001 and his Master's, also in Computer Engineering, from the same institution in 1996. He is a retired U.S. Air Officer and a graduate of the Air Force Academy. He also holds a Master's of Science degree in Chemical Engineering from UCLA. He is currently self-employed.

Dr. Jose Castro obtained his Ph.D. in Computer Engineering from the University of Central Florida in 2004. He obtained his Master's degree in Computer Engineering at the Instituto Tecnológico de Costa Rica, where he is currently the Director of the Center for Research in Computing and Professor of Computer Engineering. He holds a Bachelor of Science degree in Computer Science from the University of Costa Rica.