Design and Implementation of CITKA, a Context Based Tactical Knowledge Acquisition System

José Castro, Avelino J. Gonzalez, William J. Gerber

Intelligent Systems Laboratory
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2450
jcastro@pegasus.cc.ucf.edu, gonzalez@pegasus.cc.ucf.edu, wge@ece2.engr.ucf.edu

Abstract
The CITKA system has been developed to facilitate the acquisition of the knowledge for military tactics; this is known to be one of the limiting factors in the development of computer generated forces. This acquisition is achieved via a query session between a subject matter expert (SME) and an intelligent system. Such a system has several advantages. The most important of these would be the drastic decrease in manpower effort to implement the tactical behaviors directly into whatever language is used by the modeling paradigm. Moreover, fewer errors can be expected, making the verification of the model less difficult. Also, a tool developed to carry out the query session and build the model could serve to maximize re-use of previously defined tactics. Context-based Reasoning has been selected as the paradigm because it lends itself very well to automating the knowledge acquisition process. This paper describes CITKA, a system that develops a model of tactical behaviors via a Q&A process with the SME. A prototype was built and is described in detail. Comparisons with manual knowledge base development are being researched. The metrics used are the number of person hours required to develop the model automatically vs. manually, and the proportion of the model that can be developed in this fashion. We use a non-trivial sea scenario mission as the benchmark for the comparisons.

Introduction
It has become clear recently that the effort required to build the CGF models is both large and costly. Delugach and Skipper [1] describe an approach based on repertory grids to acquire the knowledge, and convert it into conceptual graphs. They refer to them as tracked repertory grids. They claim it facilitates knowledge acquisition, but provide no support for that claim. Other approaches involve learning directly by the agent, generally by observation. [2, 3, 4, 5]. Such learning strategies are still in an early stage of investigation, and do not provide near-term relief for the acute problem in acquiring and representing large volumes of knowledge even for relatively simple missions.

This paper describes the Context-based Intelligent Tactical Knowledge Acquisition or CITKA for short, a system developed from an investigation to semi-automatically build CGF models. The system is based on the context-based modeling paradigm called Context-based Reasoning (CxBR), which has a highly structured form and hierarchical organization that lends itself well to an automated query system, we believe this approach can reduce the effort required to build the models, as well as reduce errors. First the paper gives a brief discussion of Context-based Reasoning and the features that make it an excellent medium for the task. Latter we describe the CITKA system implementation. We end with the tests that are currently being conducted on the system to evaluate its merits.

Context Based Reasoning for Knowledge Acquisition

CxBR is based on the idea that in executing a mission, an agent will experience several different situations, all in sequence. Each situation will require certain skills and actions in order to successfully navigate/survive it.
Furthermore, situations evolve from one to the next, often abruptly, less often gradually. To successfully complete the mission, the agent has to 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 actions and procedures that properly address the current situation.
2) As a mission evolves, a transition to another set of actions and procedures may be required to address the new situation.
3) Things that are likely to happen while under the current situation are limited by the current situation itself.

CxBR encapsulates knowledge about appropriate actions and/or procedures as well as compatible new situations into hierarchically organized contexts. A sample hierarchy is depicted in Figure 1.

![Figure 1 – Context Hierarchy](image)

**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, and the constraints imposed (time, weather, rules of engagement, etc.). The **Main Context** is the primary control element for the agent. It contains functions, rules and a list of compatible next Main Contexts. Identification of a new situation can now be simplified because only a limited number of all situations are possible under the currently active context. **Sub-Contexts** are abstractions of functions performed by the Main Context which may be too complex for one function, or that may be employed by other Main Contexts. This encourages re-usability. Sub-Contexts are activated by rules in the active Main Context.

They will de-activate themselves upon completion of their actions. Refer to Gonzalez and Ahlers [6] for details on CxBR.

Therefore, in order to build a CGF model in CxBR, the following must be defined:

- The **Mission Context** must be identified, and a value set for its applicable attributes. These include name of the mission, description, weather, lighting conditions, location, constraints, and objective to be achieved in the mission.
- The Main and Sub-contexts to be used must be identified by the Subject Matter Expert (SME). This is the case whether the context is part of the original plan, or in reaction to unplanned but potentially expected situations.
- The procedures required for controlling the simulated entity while under each Main and Sub-context must be identified, specified, written and incorporated within the appropriate context.
- The rules determining context transitions must be identified, specified, written and incorporated within the appropriate context.
- The objects involved in the mission must be identified, and their capabilities specified and defined. This includes the enemy forces as well as any teammates. For example, if the mission is for a platoon of M-1 tanks, the tank’s maximum speed, turning radius, fuel capacity, weapons load, etc. is defined.
- Helping functions must be identified, specified and defined. Examples of helping functions are finding the distance between two points, or selecting the heading required for reaching a waypoint.

It is highly intuitive for an SME to provide much of this information in a query session. Of course, one cannot expect most SMEs to write the functions in a computing language. This will have to be done by a programmer or knowledge engineer. However, many of these functions will be standard, making them easily re-usable among different applications, thereby reducing the burden on the programmer. The CITKA system is based on this idea.

### The CITKA Query Process

The basis of this approach is based on an intelligent query session between the Subject Matter Expert (SME) and the CITKA system. The latter uses its own knowledge base to compose the queries in an intelligent fashion, selecting the next question based upon the SME’s previous responses. CITKA also has a feature to allow a Knowledge Engineer (KE) to complete and/or refine the knowledge entered by
the SME. However, no queries are presented to the knowledge engineer.

The process of building a CGF model begins with a specification of the capabilities of the model. This specification is mission-specific, as one would expect the assets of the task force, however, small, to differ from mission to mission. Additionally, the enemy faced may also have widely varying capabilities. Once specified, the CGF model is developed by building the context base, that is, by defining the contexts, the procedures for context actions, the rules for transitions between contexts, and the necessary objects. Once created, the context base for the CGF model of interest is incorporated within the Context-based Reasoning (CxBR) Framework and linked to the simulation of choice to be executed. The CxBR Framework is the engine that exercises the knowledge represented as a context base to achieve the desired behaviors or actions. We assume here that the CxBR Framework exists, and is already linked to the simulation of choice. Our task is to 1) specify and 2) develop the context base. CITKA addresses both tasks. We now begin with this process.

Opening Screen

The opening screen seeks to get everything started. It first asks whether the user is an SME or a KE. Figure 2 depicts the basic opening screen.

![Figure 2 – Opening Screen](image)

The user merely points to the button marked either Subject Matter Expert or Knowledge Engineer and clicks it to set the appropriate context for the session. The active button will be visually highlighted for the duration of the session, or until the user selects the other one. This will mark the user mode of the system.

CITKA User Interface Screen Design

The CITKA screens consist of three areas. See Figure 3. The first area is the control area, and is located at the extreme left-hand side of the screen, covering a narrow slit from top to bottom. This area contains the active buttons that can move the user between the different parts of the knowledge base – called the “Current Area of Interest”. The active button represents the parts of the context base currently being created. The active button will always be visually highlighted. In addition, it will also have the same set of buttons shown in Figure 2, indicating the user mode of the system. This can be either the KE or the SME. Since the more serious research issues involve providing assistance to the SME via a sequence of intelligent questions, that is by far the more interesting of the two user modes. Therefore, from this point, we will assume we are in that mode unless otherwise indicated. Lastly, this area also has a button labeled “Enter” that is to be used after the user’s responses have been entered in the response area described below.

The second area is called 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 contained here. The area is scrollable, with a scroll bar at the right hand side. It only contains text, and it is not active in any way. The third area is the response area, where the user (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 is also scrolled, with a scroll bar in the right hand side. Unlike the query area, however, the response area can be active. 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.

![Figure 3 – CITKA Screen Layout](image)

After the opening screen, the great majority of screens will adhere to the above design. The only exceptions occur when the developed part of the context base is to be reviewed. In that case, a separate window is launched to display the knowledge entered and inferred by the CITKA system, and to allow a correction if needed. We now move on to the main SME screen.
Subject Matter Expert Interface Main Menu

This screen will first allow the SME to launch a file for context base development. It has three options: 1) Initialize a new (empty) file. 2) Load another context base file used for a previously developed CGF model. This facilitates reusing elements of similar previous models. 3) Load a working file to continue its development. The response area is the prototypical windows load box.

Once the context base file has been activated, the intelligent query module of CITKA begins to ask the SME a series of queries to develop the different parts of the context base. In order, the questions will deal with the following knowledge items:

2. Other entities or objects involved in the scenario.
3. Major tasks (Main Contexts).
   3a) Actions associated with each main context.
   3b) Transitions to the next main context.
4. Sub-contexts involved with each main context
   4a) Actions associated with each sub-context.
   4b) Transitions to other sub-contexts and the associated main contexts,
5. Short/long term situational memory of relevant events/conditions, and
6. Helping functions needed to support/define abstract concepts used for context/sub-context actions, context/sub-context transition decisions or situational memory.

Some of these aspects of a context base are easier than others to capture and convert directly into useful code. More specifically, those responses that are unconstrained text will likely require interpretation by a KE for conversion into useful code. At some future point, a natural language processing module could be used to automate this process, at least partly. However, for the time being, it will have to rely on a human. 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 must turn over completion of the context base to a knowledge engineer (KE). The KE can use the Knowledge Engineering interface to complete the work, or she can utilize any other tool to do so if desired. This interface is merely a screen editor and is not described any further.

The CITKA Prototype

CITKA Design Guidelines

CITKA design specification was originally provided as a series of slide-screens that explain the interaction of the SME and the system. From the slides it clearly became apparent that the size and nature of the system demanded a developing environment that could rapidly produce a production quality system but that could also be modified easily to new requirements and specifications. This flexibility is paramount because even though the system is designed to be used without modification, given the experimental nature of the work it is very likely that on the process of testing and verification modifications will be requested.

Also the CITKA interface paradigm was considered to be sufficiently general to be applicable to any type of database backend, so even though a CxBR database is our preferable choice we have provided enough flexibility so that the design is not tied down to this paradigm, and can be easily adapted to other knowledge acquisition tasks.

There was also an interest in allowing CITKA to be available on the greatest possible number of platforms. These requirements taken together mandated the following programming language guidelines:

- Use languages that are available on the largest amount of platforms, for example, a scripting language for the definition of the Interface.
- Avoid proprietary libraries that tie down the project to a specific platform, or increase the costs of development and of deployment for potential users of the systems by requiring royalties to third parties.
- Use a language that can produce persistent data structures (that is, structures that can be easily saved to disk and loaded back into main memory) for the implementation of the database.
- Save data in text format that can eventually be easily transformed in more universal formats (i.e.: XML)
- Use a language with dynamic query capabilities.
- Preferably use a language that allows the development of expert system shells.
CITKA Delelpment tools
The guidelines allowed us to review possible candidates for implementation of CITKA, a first version of the system was developed in C++. But the final version was developed using a mix of two programming languages: CLIPS and Tcl/Tk

The CLIPS/COOL environment
Clips was chosen as the language for implementing the database for CITKA as well as a rule base that is used to execute the query of the SME. Clips is an embeddable expert system shell developed originally by NASA at the Johnson Space Center. It is available in source standard C source code and can easily be ported to any platform that has a C compiler. The COOL environment, or Clips Object Oriented Language is an extension of CLIPS that implements dynamic object oriented programming smalltalk style.

The Tcl/Tk programming language
For the interface the Tcl/Tk (pronounced Tickle/Tk) was used. This is a scripting language that can create interactive window oriented user interfaces originally developed for Unix platforms but that has been ported to Microsoft and Apple based platforms. It encourages the rapid development of production quality interfaces that are automatically ported to Unix/Windows/Mac machines.

Other languages evaluated for this purpose were Python and Java, but extensibility and ease of communication with COOL/CLIPS set the balance in favor of Tcl/Tk.

CITKA Description
The CITKA prototype system consists of four modules of independent but cooperating subsystems. These modules are:

1. Knowledge Engineering Database Backend
2. Query Rule-base Backend
3. Knowledge Engineering Interface
4. Subject Matter Expert Interface

The Knowledge Engineering Database Backend is an object base that contains the information of the Context Based CGF model. It is a data structure that holds the evolving context base, as it gradually becomes developed, either by the knowledge engineer or by the subject matter expert.

The Query rule-base Backend is a rule-based system containing the rules for executing the intelligent dialog for the subject matter expert.

The Subject Matter Expert Interface is the main interface of the system and a screenshot of it was given in the previous section. It uses the Query Rule-base Backend to define its interaction with the SME. It’s behavior is “wizard” like in so much as it is designed to guide the SME through the input screens necessary to feed the database. There is at least one rule in the Query Rule-base Backend for each input screen of the SME interface.

The Knowledge Engineering module Interface, on the other hand, can be thought of as a system administrator’s view of the system. It provides more freedom to modify and query the Context-base but does not provide any guidance in the input process. It is designed for the computer literate Knowledge Engineer that has to review, modify and expand on the information fed by the Subject Matter Expert. This module bypasses the Query Rule-base and uses the Knowledge Engineering Database Backend directly.

The requirements for the four systems are different and are treated separately in the following description.

Knowledge Engineering Database Backend
The KE database backend is central to the systems because it is the repository of the information that we are interested in gathering: the SME knowledge in a CxBR CGF model.

The knowledge engineering database backend is written in COOL. The database defines a class of object for each entity modelled in the CxBR CGF, the entities are:

- The Mission Context.
- The Main Contexts belonging to the Mission Context.
- The Subcontexts of the Mission Contexts.
- The Transition Criteria that state when a certain context should become active, or when we should transition from a context A to a context B.
- The Action Definitions of a context that indicate what should be the action to take within a context.
- Memory Variables used by the CGF model.
- Helping Functions that query states in the CGF model.
- The Entity Objects that will be used by the CGF model.

This Objects base conforms to the following constraints:

1. All Main Contexts should be part of the Mission Context (no Major Contexts should be allowed to float in the namespace).
2. Transition Functions and Action Definitions can be shared among contexts.
3. Deleting a Main Context eliminates all of it’s Sub-Contexts.
4. Deleting a Context will eliminate its Action Definitions and Transition Criteria only if they are not used by any other Context.

To implement these data structures, the native COOL class and object declarations are used. This simplifies access and maps more naturally to the Knowledge Engineering...
The KE Database Backend is written in CLIPS. This is an improvement over the first version since that one was written in C++, a very efficient language but that does not provide the flexibility and prototype turn around time that is needed for this research.

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

**Query Rule Base Back End**

During the development of the Query rule base it was found that rule activation was good for developing complex interfaces that adapt themselves to the SME needs, but that rules where overkill and cumbersome to use if the sequence of screens to implement was simple and linear.

To avoid this unnecessary complexity a scripting capability was added on top of the rule base, effectively dividing the rule base into disjoint sets that are executed sequentially. This allows CITKA to either provide on a completely linear interface where each rule base activates only one rule, on the other extreme use a single rule base that interacts intelligently with the SME to extract knowledge from her. The scripting capability was dubbed the “virtual machine” since it implements a small programming language within the CLIPS application itself.

To get a clearer picture of what this internal scripting language is we provide the following snippet:

```clips
(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, each entry corresponds to a independent rule set, “objects-get-friendly” for example, is the set of rules that query the user as to what are the friendly entities that are going to form part of the CxBR CGF model. One of the screens of “objects-get-friendly” is the screen shown as an example on the previous section.

**Knowledge Engineering Interface**

Data entry in the Knowledge Engineering Interface is provided by eight interacting dialogs. These are:

2. Main Contexts dialog.
3. Sub-Contexts dialog.
4. Entity Objects dialog.
5. Helping Functions dialog.
6. Memory Variables dialog.
7. Transition Criteria dialog.

This module is a simple data entry interface. It’s worth to note, however, that reasonable level of complexity exists, given the hierarchical nature of the database and the table nature of the required interface.

**Evaluation of the CITKA System**

The CITKA system has been evaluated for its effectiveness in [8]. There were two main issues here: 1) Estimating the reduction in person-hour effort to develop a context-based model for a particular mission, and 2) Estimating the percent of a context-based model could conceivably be automatically developed through CITKA.

In Gonzalez et al, the CITKA system is evaluated. The baseline for comparison was the model building effort involved in the Human Behavior Representation Challenge project, sponsored by DMSO in 2001. This scenario was estimated using the design specification of the CITKA system and not the system itself since the actual program was not ready at the time.

The task was to develop a model of a sea vignette and execute it in a testbed composed of SPEEDES and VRForesses. The vignette deals with a developing situation with an unfriendly maritime nation where tensions have risen progressively.

**Estimate of Effort to Develop Knowledge Base Using CITKA**

Estimates for the effort to code the Sea Vignette in CITKA where done assuming times for each of the inputs of the CITKA interface. This is obviously an estimate that is only as good as the assumptions it starts from. As of this moment, the system is being tested for consistency with these estimates with an in house subject matter expert that is using the system. Data is collected to corroborate if the
estimated projected benefit is consistent with experimental results.

Data collected corresponds to the time that the expert takes on each input screen, estimated results on the other hand are calculated on an input field by input field basis. This higher granularity of the estimate was made so as to achieve a more precise result, this is obviously not necessary for the experimental testing.

The results of Gonzalez-Gerber’s study indicate a total of nearly 71 hours assuming an SME who responds quickly, and 136 assuming a slow one.

**Estimate of Effort to Through Traditional Knowledge Base Development Methods**

By comparison, the actual time involved in developing the sea vignette knowledge base through traditional methods is shown in Table 1.

<table>
<thead>
<tr>
<th>Task</th>
<th>Hours</th>
<th>Task Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>134.5</td>
<td>Acquiring domain knowledge by KE</td>
</tr>
<tr>
<td>2</td>
<td>243.3</td>
<td>Model development, incl. algorithms</td>
</tr>
<tr>
<td>3</td>
<td>157.0</td>
<td>Learning curve on CxBR Framework</td>
</tr>
<tr>
<td>4</td>
<td>150.0</td>
<td>Coding of HBR model</td>
</tr>
</tbody>
</table>

The resulting total for all four tasks is 684.5 hours. However, making a direct comparison of this number to the estimated times obtained from the CITKA estimate requires some explanation.

Task 1 is the process of acquiring knowledge from experts. If an SME were to use CITKA in its ultimate, commercial form, this effort would not be necessary, as the SME would be doing that as part of the process of using CITKA. Therefore, it is logical to include the hours from Task 1 in the comparison.

Task 2 is directly replaced by CITKA (in its ultimate form), so it clearly should be considered. Whether to include Task 3 presents a more interesting argument. Certainly, once a KE becomes familiar with the CxBR Framework, this chunk of effort would not be required. Even with CITKA, the KE still needs to understand how Context-based Reasoning works in order to refine and complete the knowledge. However, seeing a partially complete knowledge base certainly facilitates the KE’s learning curve. Moreover, CITKA eliminates the need for the SME to understand CxBR in the first place. Therefore, arguably, this effort should also be considered in our comparison.

This leaves Task #4, coding of the model. Clearly, the actual coding of the model cannot be done by the SME, and is the responsibility of the KE. However, much of the context-based model merely represents creation of classes with the attributes assigned values, and their objects instantiated. This can easily be done by CITKA automatically, resulting in actual compilable source code. The area of greatest difficulty would be coding helping functions, and to a lesser extent, the transition rules. We can arguably assume that 80% of the code could be automatically generated (refer to section 5.3). This would replace 120 hours of the coding effort. Thus, adding together the portions of the four tasks that CITKA addresses results in nearly 655 hours. The estimated manpower effort with the use of CITKA compares very well with this benchmark number. It represents an improvement of 519 person-hours (655 – 136). It basically reduces the effort by 80%, even assuming a slow responding user. The resulting improvement is nearly 90% if a speedier user is assumed (655 – 71 = 584 hrs savings).

**Model Completion Estimate**

The next question is to estimate how much of a model could be reasonably developed automatically. This estimate is more difficult to make. At this point, we can only guess, based on our experience with the sea vignette. Let’s analyze this a bit further.

It is clear that 100% of the specification can be generated automatically. The specification consists of a listing of all defined contexts (Mission, Main and Sub-), their attributes, functions and transition rules. These are all described in textual form in the specification. Additionally, the specification includes all objects involved in the scenario, and possibly their capabilities, as either obtained from a historical database, or directly from the SME. Our estimate of 100% is justified by the fact that there is no code required with the specification.

The context base, of course, will require coding. The algorithms may be described by the SME, but they must be turned into compilable code by the KE. This would not be possible with CITKA, at least not at the level we currently envision it. However, the class definitions for the contexts and the objects that participate in the mission can be put into code directly by CITKA. So can the transition rules. Thus, it would be reasonable to expect that 50 to 80% of the model would consist of code elements that can be directly generated by CITKA. Furthermore, as some of these functions and methods are archived and reused, this total may increase. However, this number is admittedly a bit of guesswork. A more detailed analysis of this will be
forthcoming when we evaluate the prototype system currently finishing development.

Future Research
The CITKA system is at the end of it’s first development phase, it is ready for use by an SME and testing is currently being done to verify the objectives of this research. As with any software product, there are many avenues of research which have not been pursued that can enhance the usefulness of the product. Some of them are listed below

- Currently the database is stored on disk as a text file in COOL object format. This is convenient since it’s portable and easily readable and modifiable. A future enhancement would be to store the database on XML, a more universal format and rapidly becoming a standard on military and civil applications.
- The Design of CITKA specifies the definition of “Object Templates” of common objects used in the CGF definitions, this is very useful in complex scenarios where a lot of objects are involved (ie: tanks, airplanes and others). This facility is currently not provided.
- Currently the specification is not verified for correctness, formal tools available to our research lab like VSE would provide a valuable extension to the project.

Summary
We believe that CxBR, by its very nature, facilitates the knowledge acquisition task for tactical behavior models in computer generated forces. Here we described our concepts for building a system to do this very thing. A prototype has been developed to test these estimates. The prototype has desirable technical properties like multiplatform (IBM PC/Linux/Solaris/Mac) compatibility, flexible interface, incorporated expert system rule base, easily portable and readable output files.

The results indicate significant savings in development effort for models of human tactical behavior. Furthermore, in a similar analysis, we estimate that the model could be between 50 and 80% completed automatically through this system.

We expect to confirm these numbers when we finish evaluating our prototype this month. Given that SMEs are not readily available, we are using our knowledge engineers who may have become quasi-experts based on the traditional interviews conducted.

Acknowledgements
This project has been sponsored by the Defense Modeling and Simulation Office, contract number N61339-01-P-0216, LTC Eileen Bjorkman, Dr. Ruth Willis and Dr. John Tyler, Program Managers at one time or another. We appreciate their support, financial, technical and moral.

References