Comparing Two Context-driven Approaches for Representation of Human Tactical Behavior

Avelino J. Gonzalez
School of EE and CS
University of Central Florida
Orlando, FL 32816
USA

Patrick Brézillon
Laboratoire d'Informatique de Paris 6
Universite Pierre & Marie Curie, #6
75015 Paris,
France

Abstract

This paper describes an investigation that compared Context-based Reasoning (CxBR) and Contextual Graphs (CxG), two well-known context-driven approaches used to represent human intelligence and decision making. The specific objective of this investigation was to compare and contrast both approaches to increase the readers' understanding of each approach. We also identify which, if any, excels in a particular area, and to look for potential synergism between them. This comparison is presented according to ten different criteria, with some indication of which one excels at each particular facet of performance. We focus the comparison on how each would represent human tactical behavior, either in a simulation or in the real world. Conceptually, these two context-driven approaches are not at the same representational level. This could provide an opportunity in the future to combine them synergistically.

1 Introduction

Context has always played an important, if little understood, role in human intelligence. For instance, in human communications, knowing the context of a conversation precludes the need to explicitly explain several aspects of it. As an example, a discussion about skiing in Vail, Colorado in the winter strongly implies snow skiing without having to explicitly state so. On the other hand, a similar reference to skiing in Miami Beach in the summer would likely indicate water skiing.

We assert that making use of contexts can facilitate modeling of human intelligence in a computer. Here we specifically address modeling human behavior in a tactical situation. Context provides modellers of human behavior (which includes problem solving and decision making) with a very valuable tool. The recognition of its context can give an agent the means to prune the search for solutions or influence the solution selected or decision made. Historically, M^{c}Dermott [1982], M^{c}Carthy [1993], Fox et al [1983] and Guha [1991] employed context in some form or another in their pioneering work. More recently, Turner [1998], Brézillon [2005], Gonzalez and Ahlers [1993, 1998], Kokinov and Yoveva [1996] and others have conceived of approaches to modeling human intelligence that employ contexts. Other work related to formal aspects of contexts has been undertaken by Bouquet and Serafini, [2001]. There is a recent wealth of research publications presented at Context conferences, held bi-annually since 1997 in different locations throughout the world. Brézillon [1999] presents a review on context as employed in different disciplines. We take their work further in this paper by analyzing two leading context-driven knowledge representation approaches and comparing them. This is the first step in implementing and evaluating a synergistic combination of these two context-driven approaches. These approaches are Context-based Reasoning (CxBR) [Gonzalez and Ahlers, 1998] and

Contextual Graphs (CxG) [Brézillon, 2005]. We believe that this combination can be highly synergistic for modeling tactical human behavior.

The most challenging problems in modeling human behavior through contexts are:

1. Identifying a context and its relevant features - The classical Frame Problem is closely related to this issue. Once a context is identified, it is comparatively easy (although hardly trivial) to know what to do while in it.
2. Knowing when a change in context has taken place - Humans know how to do this instinctively. However, it is difficult to generalize rules that dictate when to switch to another context as a result of environmental and/or internal events.

While these approaches share many similarities, they are also quite different in how they go about using contexts. Therefore, they address the above two difficulties from different perspectives. We begin by briefly describing the two approaches and refer the reader to a host of publications that explain these concepts in far greater detail than we can in this paper.

2 Descriptions of the Relevant Context-driven Paradigms

Context-based Reasoning and Contextual Graphs are two popular and emerging context-driven approaches for representing human intelligence. We begin by describing CxBR.

2.a Context-based Reasoning

Context-based Reasoning is based on the ideas that:

1. Any recognized situation inherently defines a set of actions, procedures and expectations that properly address the situation. These sets of actions, procedures and expectations are called *contexts.*
2. As the situation evolves, a new set of actions, procedures and expectations may be required to successfully manage the newly emerging situation. Therefore, a *transition* to a new context must be effected successfully.

3. Identification of a future situation can be simplified if things that are likely to happen while under the current situation are limited by the current situation itself. This facilitates *situational awareness*.

CxBR encapsulates into contexts knowledge about appropriate actions, procedures, expectations as well as knowledge about possible new situations. By associating the potential future situations and their corresponding actions to specific situations, identification of a new situation can be simplified because only a subset of all possible situations is applicable under the current situation.

CxBR, therefore, is founded upon the following concepts:

1. Tactical experts are proficient at their task by recognizing and addressing only the essential features of a situation. Thus, they only use a small, but important portion of available inputs, and knowing what knowledge to use is based on their perceived context.
2. There are only a limited number of things that can realistically take place in any situation. This can be used to prune the search space of the problem and enhance situational awareness.
3. Contexts provide tacit expectations for individuals involved in a discourse. This permits the interlocutors to avoid describing all that is known and assumed. Problems can occur when the interlocutors are not in the same context but believe that they are.
4. Life for a tactical agent is a continual and dynamic decision making process. Decisions are made through a sequence of contexts, each of which, when active, control the agent and provide an expectation for the future during the time that each context is *active*.

5. The emergence of a new situation generally requires alteration of the present course of action. Certain environmental or internal events occur that indicate that a transition to another context is warranted.
6. Active contexts change not only in response to external events or circumstances, but also because of actions taken by the agent itself.
7. Indications that a transition to another context is warranted are relatively few for each context transition.
8. In a multi-agent situation, the active context may not be the same for all agents at the same time. This is reasonable to expect, since each may have a different situation (i.e., a different mission, different sensor inputs, different capabilities, or a different physical location).
9. Contexts are represented temporally as intervals of time rather than as time points.
10. Goals can be intervals or time points, but only to serve as transitions to other contexts.

Most tactical tasks in competitive games, warfare, as well as in many aspects of daily life consist of a pre-defined set of actions that are executed after a certain situation has been recognized. The situation could be defined by the mission, a set of orders, and a specific set of environmental conditions at that moment. The problem faced by the decision maker, therefore, is two-fold: 1) how to recognize the present situation (referred to as situational awareness), and 2) what to do when the situation is recognized (referred to as implementation of actionable information). Context-based Reasoning not only addresses the (concise) representation of the knowledge required to address the above two issues, but also defines how that knowledge is executed to achieve the desired results. Furthermore, CxBR was conceived to address efficiency and flexibility. It permits the representation of functional knowledge in many different ways. C++ methods and functions have been the preferred means of coding the desired functionality within each context. We have also successfully used neural networks within a context to accomplish tasks that were natural to a connectionist representation. See Sidani and Gonzalez [2000] and Henninger [2001] for examples of neural networks within CxBR. Other paradigms such as rule-based systems or constraint satisfaction could be easily incorporated within the CxBR architecture in applications where such paradigms are natural to the functionality being represented within a particular context.

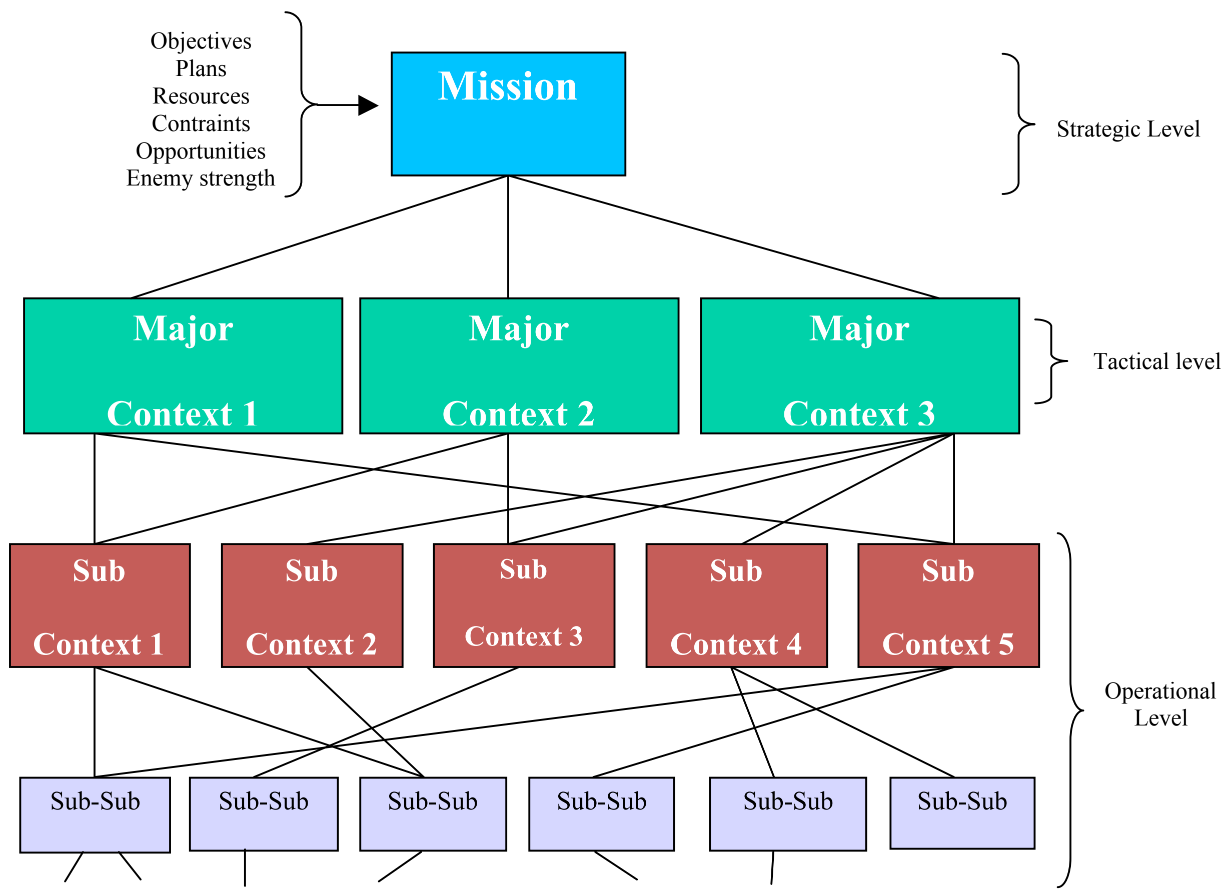


Figure 1 – CxBR Hierarchical Organization for a Particular Mission

CxBR contexts are organized as a tree, albeit possibly with multiple parents. The hierarchy consists of three levels, the Mission Context, the Major Contexts and the Minor Contexts. There is only one Mission context and it serves to define rather than control the agent. Major Contexts are the main control element, and while there may be several assigned to a particular mission, one and only one is actively controlling the agent at any one time. We refer to this controlling context as the *active context*. The Minor Contexts play a supporting role and there can be several levels of minor contexts, starting with the Sub-Contexts. Figure 1 shows an example of this hierarchy.

A full description of CxBR can be found in Gonzalez and Ahlers [1998].[1]

2.b Contextual Graphs

A contextual graph is a directed acyclic graph that represents the actions to undertake in accordance with the current context. The *action nodes* represent actions to achieve a goal while the *contextual nodes* describe the possible contextual issues of a given situation. The contextual graph is intended to represent the part of the context denoted as the *proceduralized context*, i.e. the specific context *chunks* ready to be used in the action selected. The proceduralization of the contextual knowledge makes explicit the links, especially the causal and consequential links, between contextual knowledge chunks and as such, the links become a part of the proceduralized context. Thus, the proceduralized context appears as a kind of compiled knowledge that the system will easily "decompile" to explain its reasoning. Consequently, one can regard a contextual graph as representing the proceduralized context, because for each context represented by a sequence of instantiated contextual elements, the implicit reasoning about causes and consequences implies that the action to undertake be defined without ambiguity. In other words, each sequence of contextual nodes along a branch triggers some actions as a result of rationales that are not represented on the graph but are generally known and compiled in the user's mind (proceduralized knowledge). Thus, the CxG explicitly represents the reasoning involved in the proceduralized context.

[1] The interested reader can also read further by visiting the ISL web site at http://isl.ucf.edu/CxBR_Appendix_A.doc where a detailed description of CxBR, with formal definitions, can be found.

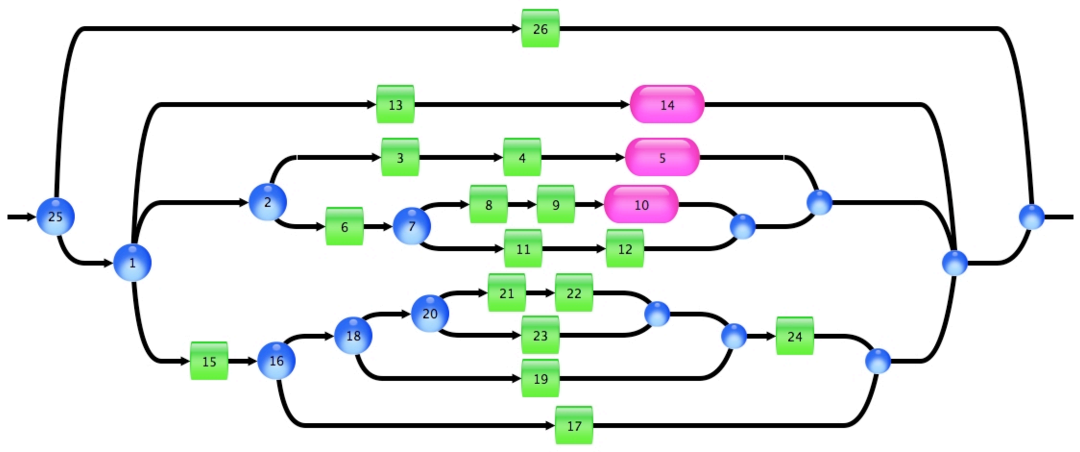


Figure 2 – Contextual Graph Organization

Figure 2 depicts a Contextual Graph and its organization of *contextual elements* (CE). A contextual element is a pair consisting of a *contextual node* and a *recombination node*. In Figure 2, the circular nodes are the contextual nodes while the square ones are the action nodes and the elongated ovals represent the activities. Note that our notation used below employs the prefix A to indicate an action and C to indicate a contextual node. The contextual nodes require a decision or an information input by the user, while the action or activity nodes perform an action or describe an action to be performed by the user. It may also modify the environment and by extension, the context. The smaller unnumbered circular nodes are the recombination nodes where a particular contextual element ends. The contextual graph is spindle-shaped in that it has one entry point and one exit point. A contextual element embeds action and activity elements included in the branches up to its corresponding recombination node. This makes it a sub-graph. Contextual elements constitute a heterogeneous set because they may not all have the same nature and granularity.

A proceduralized context is associated with a *focus*. For example the proceduralized context associated with action 8 (fourth branch on Figure 2) is the sequence of instantiated contextual elements C25-C1-C2-A6-C7. This proceduralized context can be read like:

$$(C25 = \text{no}) \;\&\; (C1 = \text{exploratory}) \;\&\; (C2 = \text{yes}) \;\&\; (C7 = \text{yes}) \longrightarrow A8$$

The purpose of contextual graphs is not to make a decision under uncertainty but rather, to represent along each branch of the graph an increasingly refined state of the problem that the operators try to identify in order to better target a solution. In dynamic application domains such as the management and operation of a subway train line, [Pasquier et al, 2000] it is fundamental to accurately represent the dynamics of the operators' reasoning. The operators seek to work with the certainty of having entered the proper branch. The chaining of the different contexts along a path forms the evolving proceduralized context. Thus, in a contextual graph, the proceduralized context evolves continuously. Some contextual elements enter while others leave the proceduralized context along the process of reasoning.

A difficulty with modeling by decision trees involves parallel sequences of actions. In the contextual graph application to subway line management, for example, the sequence in which some actions are executed is not important. Specifically, when a train must push another train that is damaged, both trains must be empty of passengers. However, the order in which each train is emptied is not important and mainly depends on the context in which each train is (e.g., at a station or not). As the contextual chunks of knowledge are too numerous to be exhaustively considered combinatorially, CxGs provide a special type of branching called *temporal branching*. Temporal branching defines several branches that may be followed independently and in any order (or in parallel). Without such a representation, one would be obliged to multiply the number of paths, one for each possible ordering to the point where the sub-sequences ending paths (after the temporal branching) are identical. Temporal branching, therefore, eliminates the need to represent the combinatorially explosive combinations of actions whose sequence of execution is irrelevant.

Figure 3 shows a simple contextual graph with temporal branching, where actions A2 (actions are represented by square boxes), A3, and A4 can be done in any sequence, but after A1 and before A5.

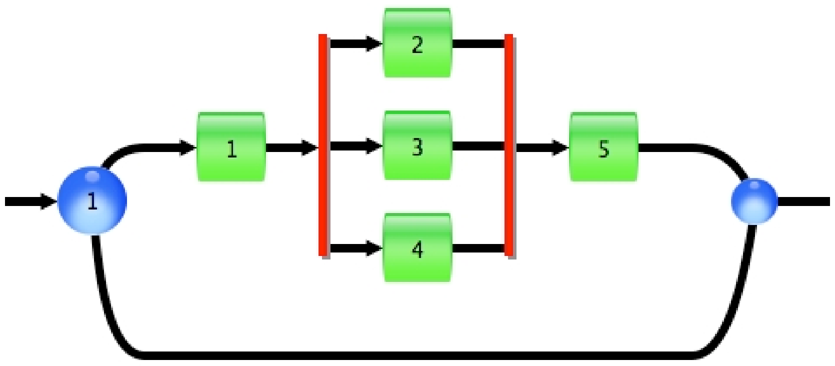


Figure 3 – Contextual Graph with Temporal Branching

Each sequence of actions in a temporal branch is locally and independently carried out. To detect such temporal branching in real applications, one has to detect sequences of actions that may actually be carried out equally well in one order or another. For example, if one sees that the operators sometimes decide to do a sequence "A-B" and other times the sequence "B-A" all under the same conditions, one can suppose that actions "A" and "B" are independent, but must both be accomplished before another step further in the graph. Such situations are easy to detect automatically from records. Pasquier [2000] and Pasquier et al. [2000] show how a series of changes leads from a decision tree as represented to a contextual graph. These changes are:

1. Identify sequences of actions that appear on several branches and combine them into a macro-action (e.g. "assembling trains" is a macro-action),
2. Merge the branches of the tree as soon as the sequence leading to the end of the incident is identical,
3. Introduce the notion of temporal branching for representing action sequences that can be executed in any order, and
4. Identify sub-graphs that appear in the contextual-graphs representation of different incident solutions.

In a contextual graph, a sub-structure may have its own significance. It can be thought as an abstracted plan. For example, in an application to subway line management [Pasquier et al, 1999] such a plan could be named "assistance to a damaged train." This plan has a goal – to push a damaged train to the end-station with another train. An explanation about exactly how to push it is included in the graph. This explanation results in the proceduralization of the context contained in a *contextual sub-graph*. This sub-graph could be found again in other contextual graphs. Such sub-graphs can be considered on their own, by operators as well as by designers. This is a chunk of knowledge that can be considered as an elementary piece of knowledge. On one hand, such an elementary sub-graph can be considered as an atomic task or a scheme of action [Pasquier et al., 2000]. On the other hand, this representation is reminiscent of Sowa's conceptual graphs [Sowa, 1984, 2000] with their mechanisms of aggregation and expansion. However, this representation is simpler than Sowa's because the contextual graph is oriented and the path provides the reasoning followed by the designer according to the different contexts. A particular solution could be found in the solutions of different incidents. Such a block corresponds to the right level of operator interpretation of events and the type of utterance between the operator and drivers of the concerned trains.

Recall that the organization of a contextual graph makes explicit the context and its dynamics for decision-making. This representation is more compact than decision trees and was accepted well by the operators in a subway line management application [Pasquier et al, 2000].

3 Commonalities among CxG and CxBR

Naturally, we found several things in common between CxBR and CxGs. The first and most obvious similarity is that CxBR and CxGs both generally attempt to emulate human-like intelligence by using context as the basic modeling element. Secondly and less obviously, both approaches are task-based. This is in sharp contrast to several popular cognitive architectures such as SOAR [Laird et al, 1987], ACT-R [Anderson et al, 1997] and others that are goal-based. These rely on the agent setting a goal, and the system trying to find a production or some other knowledge element that allows the agent to achieve that goal or sub-goal. In CxG, goals are implicit in the context being experienced. CxBR introduces its mission goals in the separate, non-controlling Mission Context, of which there is only one. The intermediate goals of the agent in the mission are defined by a plan consisting of a sequence of Major Contexts with the transition criteria defined. However, the control contexts themselves contain the functionality to perform the tasks appropriate to that context.

We have already seen that CxG and CxBR both employ the notion of context to organize knowledge. The important thing to note is CxBR and CxG are neither languages such as SOAR and ACT-R, nor true representational paradigms, but rather conceptual modeling approaches. How a model is defined or is actually implemented in either one is unspecified. This gives the developer great flexibility in adapting to the specific needs of the computing environment available to him/her. For example, CxBR could be easily modelled through Finite State Machines, Petri Nets, or Markhov Chains. The action knowledge within a Major or Minor context (Sub-context, Sub-Sub-context, etc.), can be likewise represented through functional programming, rule-based systems, constraint networks, or neural networks, among others. Transition knowledge, while traditionally represented by production rules (i.e., *transition rules*), have also been incorporated through neural networks. [Stensrud, 2005] CxGs have likewise been implemented in various different ways, such as Petri nets, Influence graphs, productions and others. It is interesting to note that while a contextual graph can be expressed fully in terms of productions, a set of productions cannot be expressed in a unique way as a contextual graph.

It is also important to note that both techniques employ "context movement" to reflect changes in the situation or in the quantity of knowledge available about a problem. This is recognition of the fact that contexts always change, and one important criterion for the success of an agent is to be able to recognize the changes in context.

4 Comparative Analysis

This section compares CxG and CxBR with respect to each of the criteria used in this analysis. We now define the criteria used in the comparison of CxBR and CxG approaches, and then analyze each approach in relation to each criterion.

4.a Criteria selection

In the previous sections we suggested that each approach has its strengths and weaknesses. We undertake this comparison for the ultimate purpose of combining the respective strengths of CxG and CxBR into one new approach that also minimizes their combined weaknesses. This new approach is not part of this investigation, but we point to it as our ultimate objective in our future work.

There are many aspects to human intelligence. We have selected a general application that influences what aspects will be used in our comparison. To this effect, we selected 10 criteria that relate to representation of tactical human behavior. These criteria were selected based on our collective experience in developing applications where human tactical behavior was being modelled. The criteria used in this analysis are:

- Class of application domain
- Context organization and granularity
- Representation of contextual knowledge
- Representation of actions
- Context movement (dynamics)
- Handling of environment
- Representation of time

- Knowledge acquisition and learning
- Representation of uncertainty and unpredictability
- Provision of explanations

We discuss each in turn below.

4.b Class of Application Domains

This metric analyses CxG and CxBR vis-à-vis the classes of problems in which they are typically employed. The class of typical application provides an insight into how these paradigms work, as well as why. This discussion also provides some foundation as to our choice of tactical human behavior as the selected general application of our investigation.

CxBR was specifically designed to model human behaviour in *tactical missions*. We define tactical missions here as complex tasks to be performed using tactical behavior in an environment. Gonzalez [2004] defines *tactical behavior* as "the continuous and dynamic process of decision making by a performing agent (human or otherwise) who interacts with and affects its environment while attempting to carry out a mission in the environment." Decisions are made at several levels during the execution of the mission. Knowledge about tactical behaviour is: [Gonzalez, 2004]

- Voluminous
- Hierarchical
- Action-based at low levels (keeping car on the traffic lane)
- Decision-based at middle levels (stop or go at yellow light)
- Decision-based at high levels (take the freeway or the back roads)
- Context dependent
- Personalized

Thorndike and Wescourt [1984] further assert that tactical knowledge can be said to be that which leads to 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. These three steps describe the very essence of these context-driven approaches.

CxBR models aim to control the physical actions of an agent situated in the real world or in a simulation, as it navigates its environment and is aware of its situation. CxBR is designed to interface tightly with the environment in which the agents will be operating. Such an environment is typically understood, but occasionally not (i.e., exploring Mars). Even in a known environment (a road network), there can be a measure of uncertainty (e.g., driving through an unfamiliar part of a city). Furthermore, tactical knowledge is inherently uncertain, as one cannot always predict the behavior of others. The interaction with the environment is generally in real time. Decisions that permanently affect the environment need to be made frequently in time and space. The agent must move through time and space, and decisions are normally, although not always, time critical. For example, an automobile driver's mission could be defined as going from her office to the airport. The environment is the road network in the city of interest, as well as any relevant elements thereof such as weather, condition of own automobile, traffic, etc.

In such applications, actions are reversible but not irrevocable. Certainly one can drive the car back to a point it has already visited (reversible action), but the original action cannot be revoked as if it never happened (irrevocable). The problems faced by the agent will have to be solved in real time, and time never goes backwards.

Tactical missions have an objective, but the goals of the agent are embedded in a plan consisting of a sequence of major contexts and transition criteria.

CxBR can be conceivably applied to higher-level tasks such as strategic or policy level tasks. However, this has not yet been verified via example.

Past successful application areas of CxBR include automobile driving [Henninger and Gonzalez, 1997; Brown, 1994; Fernlund, 2004; Gonzalez et al, 2000; Norlander, 1998], armor warfare [Barrett and Gonzalez, 2004], poker [Stensrud, 2005], submarine warfare [Gonzalez and Ahlers, 1998], Maritime navigation [Gumus, 1999], and anti-submarine warfare [Proenza, 1997].

Contextual graphs have been applied to solving general and more conventional problems in domains that are context-sensitive. Examples of this are diagnosis, interpretation, recognition and decision support. Diagnosis is context sensitive in that the same observations made under different contexts may have totally different meaning. For example, it is highly unlikely that a battery in an automobile will fail when the car is cruising at highway speeds. Even if it does fail, its consequences will be nearly nil. Thus, battery failure can be ignored while the automobile is in operation.

CxG models are designed to ask and receive answers to specific questions or make one specific decision. Often, these questions seek to further identify the context in order to make better decisions. These decisions can be made progressively as more becomes known about a problem and the context is further defined. Generally, these progressive decisions lead to one solution or to answer one question. The output of each decision leads to either a cognitive or procedural action. Decisions are generally not time-critical, and interruptions are not expected. The environment is mostly known and events predictable. However, some uncertainty can be tolerated. An agent can employ CxGs as an experience-based formalism to control its behavior and explain its actions.

CxGs have been successfully applied to provide decision support for incident management in the Paris metro system [Pasquier et al, 2000], to provide program management assistance at the US National Science Foundation [Sherwell et al, 2005], and to train automobile drivers to manage pre-critical situations [Brézillon, 2003; Brézillon and Pomerol, 2006]. There are now approximately 20 applications developed in CxGs

In summary, the two approaches address equally difficult and challenging, albeit different, application domains.

4.c Context Organization and Granularity

Contextual Graphs and Context-based Reasoning <u>organize</u> contextual knowledge. This metric seeks to identify how each approach defines such organization. It also looks at the granularity of the contexts used by models developed under each formalism. In other words, how specifically is a context being defined? This affects the level of the actions and decisions prescribed within each context. The coarser the definition of context, the more general the actions and decisions must necessarily be. We refer to these as *high-level contextual knowledge*. The reverse is also true – the finer grained a context is defined, the more specific the decisions and actions can be. We refer to this as *low-level contextual knowledge*.

Granularity here, however, does not mean who in a hierarchy of decision-makers makes the decision or carries out the action (i.e., individual, supervisor, team). Rather, it pertains to whether the context is defined in such a way as to allow a primitive or an abstract action to be executed. An example of a high-level abstract action is "Evacuate the passengers from the train". A more specific (lower-level) one is "Open the train doors, announce to the passengers that they should exit to the left, and guide them safely to the nearest tunnel exit." An even more specific one is "Press the blue button on the right side of the control panel on the train to open the left side doors. After the left side doors open, announce to the passengers to form a queue on the open doors and begin stepping out of the train car one by one and walk in the direction where the train was going, on their left side. Press the yellow button on the control panel to energize the emergency lights in the tunnel. …… ". Moving from the general to the specific requires that the context be more specifically identified. For example, the kind of train that broke down on a particular day requires that the orange button be pressed instead of the aforementioned blue button. Furthermore, the location of the train in relation to the station can dictate which doors to open and in which direction to walk after dismounting the train.

Figure 2 shows that CxBR organizes the knowledge relevant to a particular situation as a hierarchy of contexts. Major Contexts represent the gross contextual situations and are thus quite generally defined. Context is then refined though the definition of Minor Contexts. Minor Contexts can occupy several levels in the hierarchy and are designated by the precursor sub- (e.g., sub-context). Minor Contexts are used to refine the context and make knowledge available that corresponds to more specifically defined actions. There is no lower limit on this hierarchy; so theoretically, it can define contextual actions at a very fine grain. However, the contexts are defined discretely and ahead of time. A context contains all the knowledge that defines it. The contexts, while possibly of infinite levels, each incorporates a rather large chunk of contextual knowledge. Therefore, the context refinement occurs in somewhat large, discrete steps. Furthermore, as we shall see later, CxBR treats Major Contexts somewhat differently than Minor Contexts.

However, a CxBR context hierarchy only represents how the knowledge in the *context base* is defined and organized. It does not depict when it is used. For that, the *Mission Plan* and the *Context Map* are required. The Mission Plan is found in the Mission Context and provides a sequence of Major Contexts with their corresponding transition criteria that will attempt to achieve the objective designated in the same Mission Context. Of course, in most conflict-based tactical situations (e.g., games, military operations), the plan is thrown out the window almost immediately after the action begins because of unforeseen elements (the uncertainty). New plans are then drawn up, if only for temporary use. The Context Map depicts the possible transitions between contexts, as not all contexts can be reached from all other contexts.

CxGs have a well-defined process for defining contexts continuously. In other words, the context is refined discretely, but in a steady manner, one decision at a time. CxGs make no distinction about whether the context is high level or low level. It treats all contextual elements in the same way. The decisions involve which activities to invoke, and the process postpones making a decision until a maximum of information becomes known to the decision maker. There is no explicit distinction in CxG between high and low level decisions/actions. The context refinement contains small increments of contextual knowledge, making the refinement appear smooth and continuous. Figure 4 depicts an abstracted view of how the granularity of a context definition is represented in the graph of Figure 2. The inner circles reflect the most fine-grained context definitions. The outer circles represent the coarser-grained contexts. Note that fine-grained contexts are surrounded by and contained in less specific contexts. This indicates that contexts are always in relation to other contexts. Further note that while this organization has the look of a hierarchy, it is not as rigidly defined as in CxBR, providing for a more continuous context organization. One of the reasons is that contextual elements are heterogeneous in nature. Some contextual elements refer to the domain, others to the situation and yet others to the actions. Furthermore, unlike CxBR, CxG treats all levels in exactly the same fashion. Lastly, the CxG inherently describes not only the organization, but also when the knowledge is used. CxBR requires the Mission Plan and the Context Map to achieve that. The section on Context Dynamics below describes how CxGs move in and out of the context definition.

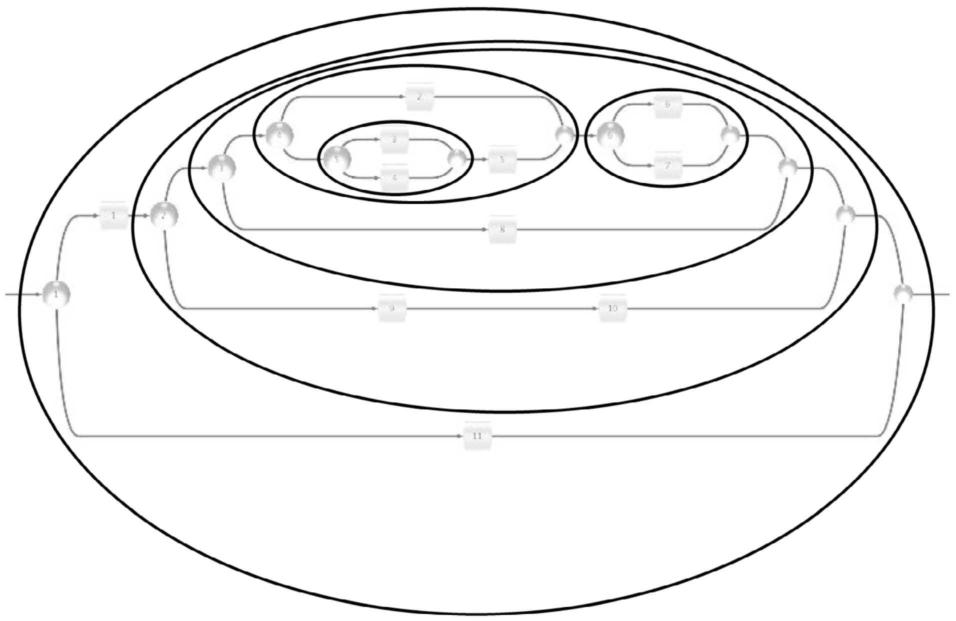


Figure 4 – Context organization in CxGs.

In summary, both paradigms represent contexts at the full spectrum of context granularity. However, by virtue of its uniform organization of contextual elements at any level of granularity, CxG is considered the finer-grained of the two.

4.d Representation of Contextual Knowledge

This metric is designed to shed light on how each paradigm represents and makes use of the knowledge within the contexts.

In CxBR, contexts are "self-contained control modules" containing knowledge that allows the agent to manage the current situation it is facing. This knowledge is composed of two general types of knowledge, and this is applicable to both, high-level and low-level contexts:

- The knowledge to manage the current situation successfully.
- The "situational awareness" to know when they are irrelevant

The first kind of knowledge refers to the action knowledge. CxBR does not specify how this knowledge is to be represented, thereby giving the developer great flexibility in how to represent this knowledge. This is discussed in the Action comparison below. The second kind of knowledge refers to context movement. The context controlling the agent at any one time is considered to be the *active* context. "Activeness" is determined largely by the environment, and is discussed further under Context Dynamics below. At the risk of over-using the high and low descriptions, we further define knowledge within each Major context as:

- Low-level knowledge: what to do within each Major context
- High-level knowledge: which Major context should be active

Contexts in CxGs are represented by the contextual knowledge known at any one time. Conceptually, a context is marked by several decisions that determine the situation to a greater or lesser specificity. The overriding objective in CxG is to postpone making a decision or invoking an action until as much as possible is known about the problem and the conditions that surround it. Specifically, contexts in CxG are defined as *contextual elements* that are identified through the contextual nodes. Therefore, the context definition is composed of:

- *Contextual elements* that embody the contextual knowledge. They include the *contextual node*, the *recombination node* and the possibly many activities located in between. Figure 2 above shows these elements of contextual graphs.
- *Proceduralized context* is that part of the contextual element/knowledge specifically used in the solution of the problem – an ordered sequence of instantiated contextual elements in a contextual graph. A contextual element enters in a proceduralized context when its value (i.e., its instantiation) is needed to describe the current state of the world.
- *Action nodes* that contain the actions to be carried out by the agent being controlled or by the human being advised. Actions are typically sequence-enforced. However, actions that can be executed in parallel are represented through *temporal branching*.

CxGs could be said to also contain the action knowledge (in its action and activity nodes) and the situational knowledge (in its contextual nodes). External knowledge is that knowledge that is not introduced in the system because it is not considered necessary up to now for problem solving. External knowledge is not considered part of the contextual graph. As in the case of CxBR, CxG does not specify how to express the knowledge within the action nodes. The representation of the problem solution remains at a high level. This, likewise, gives great flexibility to the developer.

In summary, CxBR represents contextual knowledge within the predefined modules of knowledge as action knowledge and situational knowledge. CxG represents context as a continuously evolving representation of knowledge that becomes more specific or more general depending on the situation.

4.e Representation of Actions

This metric is designed to compare how each paradigm represents actions to be executed by the agent or to be prescribed to the human user. Actions are important elements of the solution because they can represent what is to be done by the human user or by the agent to resolve the current problem or to seek additional information. As stated before in the previous section, actions can be found at any level of granularity in both CxBR and CxG.

Actions are an integral part of CxBR. Given its application class, actions are important for controlling the tactical agent. They are embedded in physical/temporal *control functions* that are inside major contexts and minor contexts. They are loosely-defined as *action knowledge*. The most common means of representing this functionality (the action knowledge) is through functional programming. However, it has also been represented with neural networks and can conceivably be represented with nearly any programming paradigm (e.g., rules, case-based systems, constraint-based search). Furthermore, there is no distinction between parallel, sequential & iterative activities – all are relegated to the control functions and handled equally well. However, CxBR represents actions implicitly – there are no boxes in the architecture representing actions. This absence of a language for representing actions has an advantage in that it introduces flexibility in how action functions are implemented, but makes control functions rather informally defined.

Likewise, activities and actions are also an integral part of CxGs. CxGs easily represent sequential activities. Furthermore, CxGs can represent parallel activities with temporal branching, (a telescoping of the representation level in different granularity) but cannot do so cleanly. CxG cannot at this time represent interactive (looping) activities. Lastly, while activities have an explicit place in the CxG architecture, how to implement activities inside these boxes is also not specified. This also introduces flexibility in how activities are represented at the expense of formality. CxG's more explicit representation of actions in the architecture is an advantage.

In summary, CxG represents activities explicitly. CxBR, on the other hand, does it implicitly.

4.f Context Dynamics

The objective of this metric is to compare how CxBR and CxG treat *context dynamics*. Context dynamics is defined as the movement among contexts. Movement can be horizontal (or lateral) – leaving one context and entering another of equal level. Movement can also be vertical – specializing the context more and more as more information is obtained. With respect to an evolving focus, movement can be considered as context evolution. The context can also be generalized, as activities are completed.

In CxBR, context is dynamic because it changes as the situation changes. One major context is always designated as "*active*" and therefore in control of the agent. Other major contexts are "*inactive*" but always hoping to become active. Contextual movement occurs through the transition knowledge, most often implemented as *Transition Rules* that monitor the environment for situational criteria that signal a change in situation that warrants a change in context. Transition rules are context-sensitive and only monitor the environment when the major context to which they are associated is active. Transition rules are in fact a rule-based system, and this is the most popular means of implementing transition knowledge. However, they have been implemented as conditional statements within control functions in some applications. Neural nets have also been used to execute these transitions [Stensrud, 2005].

Making the right transition implies being aware of the current situation to the fullest extent permitted by sensory limitations. This is the situational awareness mentioned earlier in this paper. Environmental criteria typically determine which context is active in CxBR. This is true most significantly at the major context level, but also true at the minor context levels.

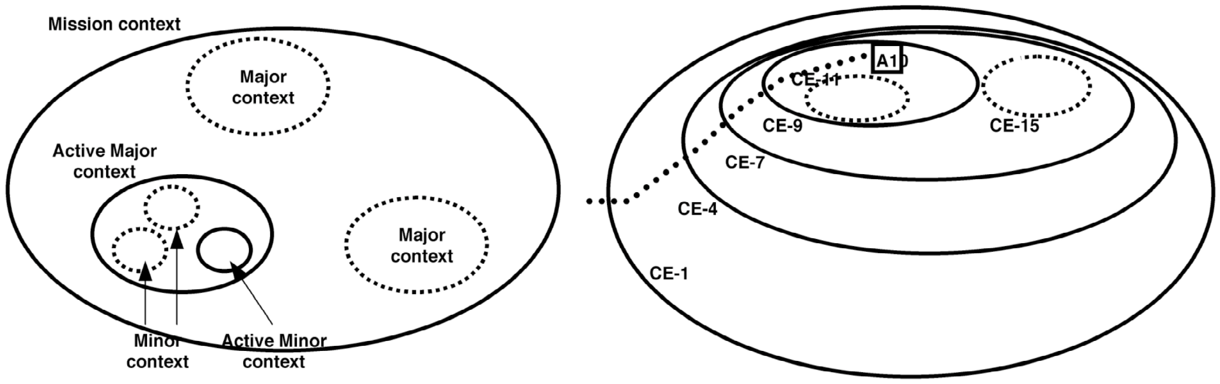
Context dynamics in CxBR is discrete. Context shifts occur discretely and definitively. The dominant, well-defined movement is among several peer modules of equal importance – major contexts. This is referred to as *horizontal* motion. Therefore, CxBR represents lateral context shifts quite well. *Vertical* shifts – the further specification of the context – are done through the context hierarchy. This refers to the minor contexts, i.e., sub-contexts, sub-sub-contexts, etc. This can be considered a refinement of the higher-level context. Knowledge in the form of *action rules* within the major contexts determines when the minor contexts should be activated and de-activated. Vertical movement takes place through a process similar to the context activation. In fact, minor contexts are said to also become active within an already active higher-level context. This happens through a process very similar to the transition criteria except that it occurs strictly within the higher-level context that contains the minor context being activated. Although they have a place in the architecture, CxBR does not specify how minor contexts should be expressed. Therefore, vertical movement is not well defined.

In CxGs, context is dynamic because as further contextual elements enter the proceduralized context, context definition becomes more specific. The part of the contextual knowledge that becomes highly relevant according to the context is the proceduralized context (PC). This is the part of the contextual element that becomes instantiated depending on how the contextual node directs the focus of the execution of the CxG. The proceduralized context is in effect, a path through the contextual graph that represents the actions executed or prescribed as well as the decisions made by the contextual nodes. The PC cuts a path through the contextual elements at all levels. We refer to this as the *vertical* motion. As contextual elements are exited throughout the PC at the recombination nodes, context definition now becomes more general. This movement, although technically discrete, is in such a fine grain that it can be considered to be nearly continuous rather than discrete. Contexts continue to be defined at greater or lesser specificity as contextual elements are entered and exited. Therefore, CxG can be confidently said to move well vertically. Horizontal movement occurs along a path when one leaves one contextual element at its recombination node and enters the next contextual element as directed by the decision point in the contextual node. However, in CxG, the agent's context is not a refinement of the group's context and thus there is no hierarchy among contexts.

A CxG is a heterogeneous population of contextual elements of unequal granularity that are ordered in sequence. CxGs, therefore, have no natural way to move horizontally between contexts of similar level because paths from one contextual node are exclusive. If the situation changes radically so that a shift in the high level context becomes necessary, CxG would handle that by shifting to another contextual graph. In the onion metaphor [Pasquier et al, 2000], that would mean peeling a different onion. However, this operation has not been considered relevant in past applications of CxGs.

In summary, the best defined movement in CxBR is lateral movement. Vertical movement is defined hierarchically, and is thus different than horizontal movement. CxGs handle mainly one type of movement - vertical. This is an advantage for applications that do not require change of high-level contexts, but a disadvantage otherwise.

Figure 5 best shows the contrast between these two approaches. Figure 5a shows the graphical representation of a CxBR operational layout (as opposed to the hierarchical representation of Figure 1). Figure 5b does the same for Contextual Graphs.



(a) **(b)**

Figure 5 – Operational representations of CxBR (a) and CxG (b) approaches vis-à-vis their granularity

4.g Interaction with the Environment

This metric seeks to evaluate the type of environment with which each paradigm is better at interacting, and how each paradigm interfaces with its environment.

Of course, interaction with its environment is a critical aspect of tactical missions. Therefore, CxBR agents that display tactical knowledge must be highly interactive with the real or simulated environment in which they are situated. In general, this environment is unpredictable, and complete information about the environment is generally not available. Furthermore, the decisions of the agent change the environment in some way.

In most CxBR applications, the interface with the environment is done through a fact base that reflects the world as seen by the agent. There are typically two types of fact bases: a local fact base that reflects only those things about the environment perceivable by the individual agent, and a global fact base that reflects things about the environment that are known by all agents (e.g., weather, the score, the time).

In typical CxG applications, a real-time environment is generally not necessary. While agents can certainly interface with a real-time environment for some applications, in others, the system interacts with a human user. CxGs can and do operate in applications domains where the human operator is under temporal pressure to make a decision. However, there is no explicit representation of real time. These applications reflect decision support and their output is advice on how to proceed. Part of contextual knowledge can also be acquired directly by sensors (GPS, mouse clicks in web, etc.). Once the problem is solved, the task is finished. Therefore, we can say that interaction with the environment is essential to CxBR, but not always to CxG.

For CxGs, a change in the environment may have two cases. If the change is expected, the actor enters the appropriate focus and the change is thereafter irrelevant. If, on the other hand, the change is unexpected, the actor must enter a new CxG. This is a quite similar process to context transition in CxBR.

In summary, interfacing with a real time environment is critical for CxBR, but not so critical for CxG in their typical applications. CxG applications normally interact with a human user who may be the intermediary to the real world. Understanding and managing the environment presents bigger challenge in typical applications of CxBR.

4.h Representation of Time

This metric seeks to compare how CxBR and CxG handle time.

In CxBR, time is inherently represented explicitly - a tactical mission develops over time, so time is inherent therein. Functions can be written to make use of time as a variable. However, it can also be represented implicitly if desired by the modeller. Furthermore, major contexts are active for a time period, and then become inactive. Lastly, the environment, whether simulated or real, by necessity, incorporates the concept of real time. While CxBR is best at handling time on a real, continuous basis, it can also be handled, at least conceptually, on a discrete-event basis.

Time is implicitly represented in CxGs. Ordering actions and activities in CxG is the unique way to represent time in CxGs. The user/agent comes in and out of contextual elements over time, but this is not necessarily real time. Furthermore, all time must be occupied by some activity – a delay is considered a waiting activity. CxG handles time equally well on a continual or discrete basis.

In summary, CxBR and CxG both handle time well implicitly. Furthermore, CxBR can handle it well explicitly.

4.i Knowledge Acquisition and Learning

Learning is a very important aspect of human behavior. This metric seeks to compare how they incorporate learning and knowledge acquisition. The use of contexts as the basis for modeling human intelligence provides several advantages in learning. If the context in which something is to be learned can be identified, the learning process can be made effective as well as efficient. This is because being situated in a context already implies a certain amount of knowledge that the learner does not have to discover. The more specific the context, the more knowledge is already implicit.

Other than the fact that it employs contexts and that context facilitates learning in general, CxBR has no natural means to acquire knowledge or learn. Several approaches to learning/knowledge acquisition in CxBR have been devised in prior research that has yielded promising results. These include:

- CITKA: A semi-automatic query system to gather knowledge from experts directly [Gonzalez et al, 2006]
- GenCL: To build a context base automatically from observation of human behavior in simulations [Fernlund, 2004]

- Reinforcement CxBR: To use reinforcement learning to refine CxBR models. [Aihe and Gonzalez, 2004]

However, these are not integrated into the CxBR paradigm itself.

CxGs, on the other hand, excel in this aspect. CxGs can easily accept new (contextual) knowledge from the *external knowledge* (i.e., the knowledge not directly associated with the current focus) when a contextualized procedure results in failure during problem solving. This can be used to build new paths incrementally through the contextual knowledge. This process can be likened to reinforcement learning, except an expert must provide guidance, and it cannot learn totally unknown knowledge at this time. This is because it has no way to automatically obtain feedback from the environment as to whether any new actions were successful or not. Incremental acquisition of new knowledge and learning of a new practice are embedded in the tasks represented in CxGs. Learning, nevertheless, is clearly a strength of CxG.

4.j Uncertainty and Unpredictability

This is a comparison of how each paradigm views and is able to handle uncertainty. The applications to which CxBR is naturally associated typically contain uncertainty as well as unpredictability. Uncertainty can come from the environment as well as from the knowledge itself. Unpredictability can come from events in the environment. Lastly, all information may not be available at the time the decision is to be made.

Conceptually, CxBR is naturally adaptable for handling uncertainty and impreciseness within its Transition Rules. Whereas transition to another major context is a crisp decision, it can be made by accounting for the relative weight of potentially contradictory information. This is particularly true in military encounters when the so-called "fog of war" presents conflicting information to the decision-making agent. Methods that manage uncertainty and/or impreciseness, such as fuzzy logic could be used to represent the uncertainty/impreciseness and make a crisp decision. However, current implementations of CxBR have not incorporated uncertainty or impreciseness, so this hypothesis has not been validated at this time.

CxGs, on the other hand, do not naturally handle uncertainty. This is because the decisions are purposely postponed until all information is available. In a non-real time environment, this is feasible, desirable and practical to do. If the human user does not have the answer, he/she must then seek it. Thus, contextual nodes are inherently crisp. Nevertheless, CxG can incorporate some impreciseness or uncertainty by discretizing levels of certainty/precision in contextual nodes. Unpredictability is somewhat irrelevant in the domains where CxG are applied. Handling uncertainty and unpredictability, however, is not one of CxGs strengths at the moment.

4.k Providing Explanations

Explanations are becoming more and more important for ensuring user confidence in an intelligent system's outputs. Explanations can be considered the reverse of learning – they teach the user about the computer's knowledge. The purpose of this criterion is to evaluate the ease with which explanations can be provided by each paradigm.

Explanations are not considered critical in tactical mission domain, other than possibly for after-action review of a trainee's performance. Therefore, explanations in CxBR have not been explored. Explanations could be conceivably generated quickly and easily by displaying the major and minor contexts active at any one time – in real time, or in a chronological history. This explanation could include the transition rule that fired to activate the currently-active context and/or any context that had previously been active. However, CxBR cannot naturally handle explanations at a finer granularity than this.

CxGs, on the other hand, handle explanations exceptionally well. CxGs use the context definitions to provide explanations. In the same manner that we spoke of learning being facilitated when in context, explanations are also facilitated by knowing the context in which an action was executed or prescribed, or a decision was made, or when and why a contextual element was acquired by the system. The information about the context is put in at the time of the creation of the context in the graph, but it does not represent a burdensome overhead to the developer in terms of extra definitions. The knowledge acquisition system also records who entered the information and when, making deeper explanations possible by tracing the source of the knowledge to the human developer.

In summary, explanation is clearly a strength of CxG.

4.l Summary

In summary, CxBR and CxG do not work at the same level of granularity. While this may at the surface seem to be a negative aspect, in fact it is a strength because a future integration of the two approaches promises to become more powerful in its representation of real world problems. Specifically, CxBR through its integration with CxG, could be useful in domains other than tactical human behaviour representation.

In conclusion, CxBR is better at doing some things and CxG is better at doing others. One question is what does "better" mean? Short of a quantitative measuring stick that would be difficult to define, we qualitatively analyzed the two approaches in light of each of the 10 criteria. The main consideration when determining which was better at doing something, however subjective, was how easy or cumbersome would it have been for a developer to incorporate the aspects of human intelligence reflected by each criterion. A "natural" means of incorporating important features spoke well of the formalism that exhibited it. Lack of ease in implementation made it undesirable in this particular entry. Table 1 below shows a summary of the evaluations done in this chapter.

Table 1 – Summary of the CxBR and CxG comparison (to be placed in this general vicinity)

Furthermore, based on what we have discussed, a cross mapping of terms can now be done to better represent the similarities. This is shown on Table 2.

Table 2 – Terminology Cross reference between CxBR and CxG (to be placed in this general vicinity)

5 Synergistic Integration of CxG and CxBR

It is clear from the above analysis that CxG and CxBR each does some things well and others less effectively. A cursory look at the analysis reveals that in general, one's strengths are the other's weaknesses and vice-versa. This provides a unique opportunity to combine the two synergistically into one approach that builds on the strengths of each paradigm.

Specifically, this new paradigm could employ CxBR's hierarchical architecture to represent the high level knowledge, and use the refined context advantages of CxG to define the lower-level knowledge within a context. CxGs could also be used to represent minor contexts when their actions are complex. Furthermore, the learning and explanation capabilities of CxG would be advantageous in this new paradigm.

The new architecture, tentatively labelled *Context-driven Modeling Architecture for Human Intelligence* (CDMAHI) could look as follows for a major context. In the concept depicted in Figure 6, a CxG is used to organize the action knowledge within a major (or even a minor) context. This would make use of the ability of CxG to offer fine grained organization within a body of knowledge. The CDMAHI paradigm retains the horizontal as well as hierarchical mobility associated with CxBR and enhances the internal organization of the action knowledge within the contexts. A CxG here represents a decision making step that is applied each time one applies the major context, i.e. each time a contextual element of this context is modified.

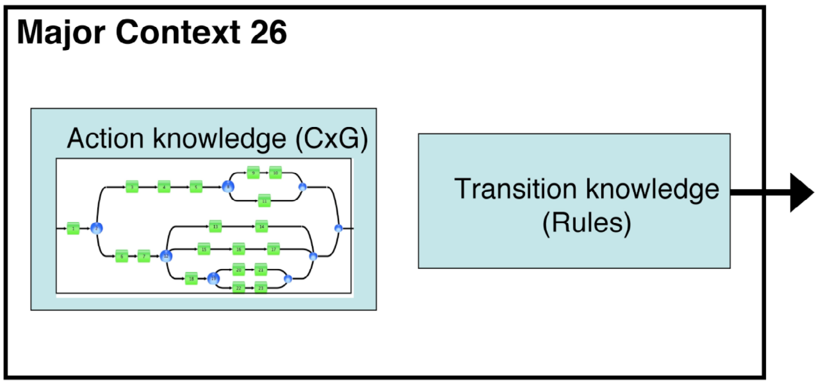


Figure 6 – Representation of a Major Context with Contextual Graph as transition knowledge

This arrangement would eliminate the potential for an explosion in the number of contexts, where different values to certain critical elements of the environment could cause new contexts to be created. Another advantage the new architecture could have is that learning could be facilitated wherever the CxGs are employed. This would be specially the case when the system must interact through a human user. It could be less so in autonomous agents. The explanation feature of CxGs could be used to advantage in applications of agents to training.

The contemplated integration and its application are best described through the following example.

6 Example

In order to show how CDMAHI can be applied to represent a common problem, we define a vignette involving a mission called **Search-and-Destroy**, where an armor platoon (four tanks) is tasked with seeking enemy forces and destroying them after reporting their location to their field headquarters. The platoon is given a sector of territory to search. This hypothetical territory is marked by four GPS points and includes two small villages, a river and three dense forests. Figure 7 depicts an artist's conception of this sector in a plan view. The sector along with all objects therein, the weather, and the orders represent the environment with which our agents must deal.

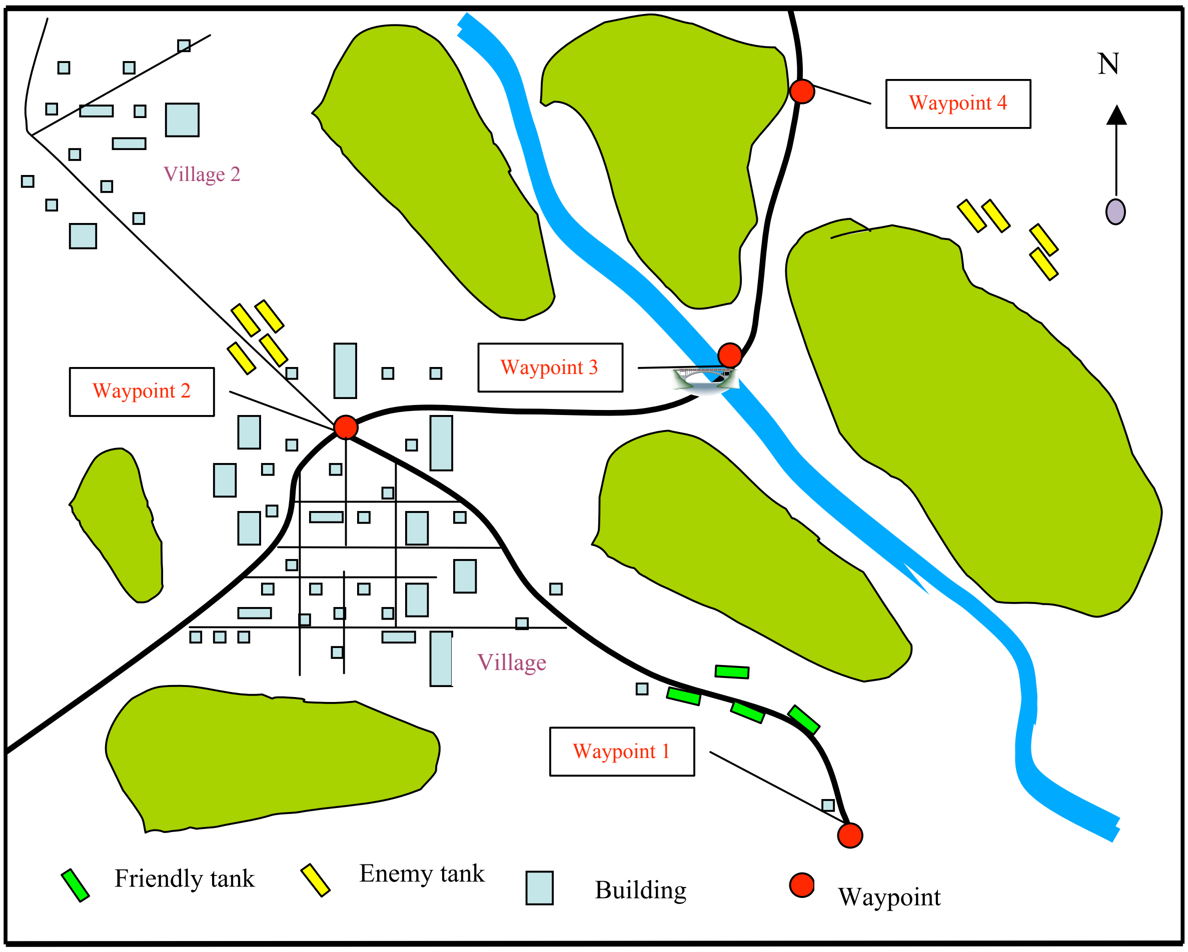


Figure 7 – Sector of Territory of Interest

The mission is to proceed from the starting point of Waypoint #1 to the final point of Waypoint #4 and in the process, seek enemy contact to destroy them. There are reports of enemy activity in the sector. Their exact whereabouts are unknown (an unknown environment), but they are for certain north of Waypoint 1. This is given as certain information. The task force will have at its disposal several contexts with which to manage the mission. The major and minor contexts assigned to the mission context for this agent to employ are the following:

1. **RoadMarch**: Movement without regard for enemy contact.
2. **SeekContact**: Move in a cautious fashion, using bounding overwatch when feasible, and in diamond formation when it is not. **BoundingOverwatch** and **DiamondFormation** are sub-contexts attached to this major context.
3. **MovementToContact**: Enemy has been spotted and the platoon move towards an attacking position. Includes the **SeekAdvantageousTerrain** sub-context.
4. **Attack**: Begin firing. Includes **AssignTarget,** and **ZigZagApproach** sub-contexts.
5. **Withdraw**: A force stronger than them has been met and must break contact. **SeekShelter, SeekHiding** and **ProvideVisualCover** are available sub-contexts
6. **EscapeAmbush:** The agents have been ambushed and must decide what to do to survive it. Includes **FrontalAttack**, **EnemyBehind** sub-contexts.

The platoon is considered as one agent acting in cohesion for the purposes of simplification. The initial plan for the agent platoon is to employ the major context **RoadMarch** between Waypoints 1 and 2, since scouts in the vicinity report the area free of enemy activity. Upon reaching Waypoint 2, the agent is to transition to a **SeekContact** context. Given the open terrain around it, they will employ a **DiamondFormation** sub-context for quick travel in potentially dangerous areas, but with low potential for ambush. If an enemy is sighted at any time in between, the agent platoon is to immediately transition into **MovementToContact** and prosecute an attack on the sighted enemy. When it reaches the bridge at Waypoint 3, the platoon will discover that the terrain lends itself to being ambushed, so before crossing the bridge it is to shift to a **BoundingOverwatch** sub-context that will protect them better against an ambush. They will continue in this mode until they make contact with the enemy or reach Waypoint 4.

Upon embarking on the mission, the platoon finds enemy presence just beyond Village 1. They move to contact the enemy, attack them, and destroy the enemy. After suffering only light damage and no casualties, they proceed towards the bridge. They cross the bridge and fail to detect another enemy platoon hiding on the far side of the large forest. The enemy gets behind them. The first shot announces the presence of the enemy, and they proceed to transition to **EscapeAmbush**, with the sub-context of **EnemyBehind**.

The Enemy-Behind sub-context can be represented by a Contextual Graph. This is shown in Figure 8.

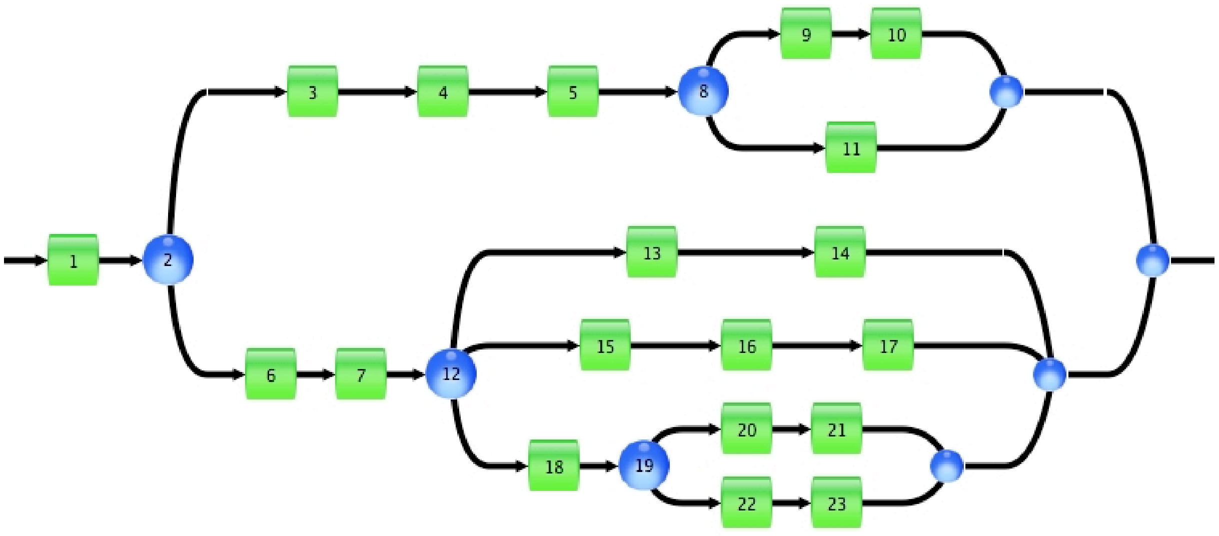


Figure 8 – Contextual Graph depicting the EnemyBehind sub-context.

The following legend describes the meaning of the contextual nodes and actions.

```
1: Detection of enemy behind
2: Has the enemy seen us?
        no      3: Check if it can see us
                4: Inform Company Commander
                5: Collect information
                8: Have we a chance to win?
                        yes     9: Plan the attack
                                10: Attack enemy
                        no      11: Withdraw rapidly
        yes     6: Inform Company Commander
                7: Collect information
12: What is the enemy doing?
        escape 13: Plan the attack
```

```
          14: Attack enemy
attack    15: Find a secure location
          16: Prepare the defense
          17: Request an air strike
observe
          18: Assess the enemy's advantage
          19: Can we win?
                 yes    20: Plan the attack
                        21: Attack enemy
                 no     22: Request an air strike
                        23: Withdraw rapidly
```

In this example, the CxG embedded in a Sub-context is able to look at the situation and determine which course of action to take in response the newly-discovered enemy. This course of action is represented as action nodes in the contextual graph of Figure 10, but could easily be made as transition rules to activate other major contexts.

7 Summary and Conclusions

This investigation has analyzed the Contextual Graph and Context-based Reasoning, paradigm to better understand them and draw some parallels as well as distinctions. It compared how each treats aspects of human intelligence and behavior. The investigation provided several deep insights into contextual reasoning in general.

It is clear from the research that, in spite of some key but general commonalities, these two approaches are quite different in how they represent problems as well as the kinds of problems they address. Their treatment of contexts, however, is somewhat similar, and most importantly, synergistic. Therefore, we hope that the results reported here will someday lead to a new approach where the best features of CxBR and CxG are integrated into one. This notional approach is briefly explained and then an example in tactical behaviors is presented. This is the type of application commonly associated with CxBR. Future research plans includes the implementation of this vignette into a prototype system, and evaluate the improvement in performance of the resulting system with one implemented solely in each of the two approaches.

8 References

Aihe, D and Gonzalez, A J (2004). "Context-driven Reinforcement Learning", Proceedings of the Second Swedish-American Workshop on Modeling and Simulation, Cocoa Beach, FL, February2-3.

Anderson, J R, Matessa, M and Lebiere, C (1997). "ACT-R: A theory of higher level cognition and its relation to visual attention." *Human Computer Interaction* **12** (4) pp 439-462.

Barrett, G C and Gonzalez, A J (2004). "Expanding Knowledge Representation within Context Based Reasoning to Facilitate Modeling Collaborative Behaviors." Proceedings of the European Simulation Interoperability Workshop, Euro-SIW, Edinburgh, Scotland.

Brézillon, P (1999). "Context in problem solving: A survey." *The Knowledge Engineering Review* **14** (1) pp 1-34.

Brézillon, P (2003). "Representation of Procedures and Practices in Contextual Graphs." *The Knowledge Engineering Review* **18** (2) pp 147-174.

Brézillon, P (2005). "Task-realization models in Contextual Graphs." *Modeling and Using Context (CONTEXT-05)*, A. Dey, B. Kokinov, D. Leake, R. Turner (Eds.), Springer Verlag, LNCS 3554, pp. 55-68

Brézillon, P, Brézillon, J and Pomerol, J-Ch (2006). "Decision making at a crossroad: a negotiation of contexts", Proceedings of the Joint International Conference on Computing and Decision Making in Civil and Building Engineering, pp. 2574-2583.

Brown, J (1994). "Application and Evaluation of the Context-based Reasoning Paradigm." Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, July.

Fernlund, H (2004). "Evolving Models from Observed Human Performance." Doctoral dissertation, Department of Electrical and Computer Engineering, University of Central Florida, Spring.

Fox, M S, Kleinosky, P and Lowenfeld, S (1983). "Techniques for Sensor-based Diagnosis." Proceedings on the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, Germany.

Gonzalez, A. J. (2004). "Presentation to faculty at Air Force Institute of Technology." December, Wright-Patterson Air Force Base.

Gonzalez, A J and Ahlers, R H (1993). "Concise representation of autonomous intelligent platforms in a simulation through the use of scripts." Proceedings of the Sixth Annual Florida Artificial Intelligence Research Symposium, Ft. Lauderdale, FL, April.

Gonzalez, A J and Ahlers, R (1998). "Context-based Representation of Intelligent Behavior in Training Simulations." *Transactions of the Society of Computer Simulation* **15** (4) pp 153-166.

Gonzalez, F G, Grejs, P and Gonzalez, A J (2000). "Autonomous Automobile Behavior through Context-based Reasoning, Proceedings of the International FLAIRS Conference, Orlando, FL, May.

Gonzalez, A J, Castro, J and Gerber, W E (2006). “Automating the Acquisition of Tactical Knowledge for Military Missions. *Journal of Defense Modeling and Simulation* **3** (1) pp 145-160.

Guha, R V (1991). "Contexts: a formalization and some applications", MCC Technical Report ACT-CYC-423-91 December.

Gumus, I (1998). “A Threat Prioritization Algorithm for Multiple Intelligent Entities in a Simulated Environment.” Master’s Thesis, Department of Electrical and Computer Engineering, University of Central Florida, summer.

Henninger, A E (2001). "The Use of Neural Network Based Movement Models to Improve the Predictive Utility of Entity State Synchronization Methods for Distributed Simulations." Doctoral Dissertation, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL, Spring.

Henninger, A E and Gonzalez, A J (1997). "Automated Acquisition Tool for Tactical Knowledge." Proceedings of the 10th Annual Florida Artificial Intelligence Research Symposium, May, pp. 307-311.

Kokinov, B and Yoveva, M (1996). “Context Effects on Problem Solving.” In Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ:Erlbaum.

Laird, J E, Newell, A and Rosenbloom, P S (1987)."Soar: An Architecture for General Intelligence." *Artificial Intelligence* **33** (1) pp 1-64.

M^{c}Carthy, J (1993). "Notes on Formalizing Context." Proceedings of the 13th IJCAI, **1** pp 555-560.

McDermott, J (1982). “R1: A Rule-based Configurer of Computer Systems.” *Artificial Intelligence* **19** (1) pp 39-88.

Norlander, L (1998). “A Framework for efficient Implementation of Context-Based Reasoning in Intelligent Simulations.” Master’s Thesis, Department of Electrical and Computer Engineering, University of Central Florida, 1998.

Pasquier, L (2000). « Raisonnements basés sur le contexte: Contextes proCéduralisés, graphes contextuels et schèmes d'action. » Research Report LIP6 N.2000-010, Université de Paris 6, France.

Pasquier, L, Brézillon P and Pomerol J-Ch (2000. “From representation of operational knowledge to practical decision making in operations.” Decision Support through Knowledge Management. S. Carlsson, P. Brezillon, P. Humphreys, B.G. Lundberg, A. M^{c}Cosh and V. Rajkovic (Eds.). Akademitryck AB, Edsbruk, Sweden, ISBN 3-901882-11-1, pp. 301-320.

Proenza, R (1997). “A Framework for Multiple Agents and Memory Recall within the Context-based Reasoning Paradigm.” Master’s Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Spring.

Sherwell, B W, Gonzalez, A J and Nguyen, J (2005). “Contextual Implementation of Human Problem-solving Knowledge in a Real-World Decision Support System.” Proceedings of the Conference on Behavior Representation in Modeling and Simulation, Los Angeles, CA, May.

Sidani, T A and Gonzalez, A J (2000). "A Framework for Learning Implicit Expert Knowledge through Observation." *Transactions of the Society for Computer Simulation* **17** (2) pp 54-72.

Sowa, J F (1984). *Conceptual structures information processing in mind and machine*. Reading, MA: Addison-Wesley Publishing Company.

Sowa, J F (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks Cole Publishing Co.

Stensrud, B S (2005). "FAMTILE: An Algorithm for learning High-Level Tactical Behavior from Observation." Doctoral Dissertation, Department of Electrical and Computer Engineering, University of Central Florida, May.

Thorndike, P W and Wescourt, K T (1984). "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.

Turner, R M (1998). "Context-Mediated Behavior for AI Applications." In Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98, Vol. 1, pp. 538-545, June 1-4, Castell, Spain.

Table 1 – Summary of the CxBR and CxG comparison

Criterion	CxBR	CxG	Comment
Scope of application			Significantly different
Granularity			CxG more fine grained decisions
Representation of Activity		CxG better	CxG more explicit
Repr. of context dynamics	Discrete	Continuous	Horizontal vs. Vertical
Environment interface	CxBR better		Critical to CxBR
Representation of time	CxBR better		Explicit and implicitly
KA and learning		CxG better	Natural means of learning
Uncertainty/impreciseness	CxBR better		
Explanation Provision		CxG better	

Table 2 – Terminology Cross reference between CxBR and CxG approaches

CxBR	CxG
Transition rule	Contextual node
Action knowledge	Procedures and practices
(mission context + sublevels)	Contextual Knowledge
~(mission context + sublevels)	External knowledge
Activeness Sequence (all context levels)	Proceduralized Context
Transition criteria	Focus
Inactive contexts	Non-proceduralized contextual knowledge