EXPLANATIONS IN CONTEXTUAL GRAPHS: A SOLUTION TO ACCOUNTABILITY IN KNOWLEDGE BASED SYSTEMS

by

BRIAN SHERWELL
B.S. University of Central Florida, 2002

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2005

ABSTRACT

In order for intelligent systems to be a viable and utilized tool, a user must be able to understand how the system comes to a decision. Without understanding how the system arrived at an answer, a user will be less likely to trust its decision. One way to increase a user's understanding of how the system functions is by employing explanations to account for the output produced. There have been attempts to explain intelligent systems over the past three decades. However, each attempt has had shortcomings that separated the logic used to produce the output and that used to produce the explanation. By using the representational paradigm of Contextual Graphs, it is proposed that explanations can be produced to overcome these shortcomings. Two different temporal forms of explanations are proposed, a pre-explanation and a post-explanation. The pre-explanation is intended to help the user understand the decision making process. The post-explanation is intended to help the user understand how the system arrived at a final decision. Both explanations are intended to help the user gain a greater understanding of the logic used to compute the system's output, and thereby enhance the system's credibility and utility. A prototype system is constructed to be used as a decision support tool in a National Science Foundation research program. The researcher has spent the last year at the NSF collecting the knowledge implemented in the prototype system.

ACKNOWLEDGMENTS

It is noted that the work to develop an API for Contextual Graphs and the programming of the prototype system, without the explanation faculty, was co-developed with Johann Nguyen. Johann is a graduate student in the Department of Electrical and Computer Engineering at the University of Central Florida.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

"Verification of a Knowledge Based System (KBS) is not enough, KBSs need to justify and be accountable for their predictions," (Richards 2003). In this statement, Richards, sets forth the idea that a system must do more than just give a correct output. A system must also be able to explain how it arrived at the answer.

Explanations are the most common method used by humans to support decisions (Schank 2003). When a system explains its reasoning and actions, it allows the user to gain a sense of understanding. "But the purpose of explaining is not only a technical one. The (human) user is also interested in how much trust he or she can have in a system. An obvious approach to increasing the confidence in a system's result is to output explanations as part of the result." (Roth-Berghofer 2004). This allows for the user to ascertain how much trust she can have in the system.

Trust is an important component for any system relied upon to make important decisions. It is human nature to question decisions produced by other humans, let alone decisions produced by a machine. One way to increase this trust in the system is by showing how the system arrived at a specific decision. If a user is able to see how the system derived the answer to their query and is able to follow that line of reasoning, she will be more likely to trust the system in future encounters (Herlocker, Konstan et al. 2000).

Explanations have been incorporated into intelligent systems since the 1970s. The early efforts of explainable AI centered largely on the knowledge representation paradigms of rule-based and case-based reasoning. The explanations produced by these systems were effective for explaining the output, but not for explaining how the system

arrived at its decision. The way in which these systems represented their knowledge made producing explanations of this nature difficult. Intelligent systems needed to incorporate separate rule bases or entire intelligent systems to decipher the results of the first system. Therefore, to produce an explanation that is efficient and an accurate representation of how the system behaves the knowledge must be represented in a way that facilitates the process. By building an intelligent system using Contextual Graphs, the knowledge can be represented in a manner that is easily explained.

The Contextual Graph (CxG) formalism offers a method of representing knowledge in a structured approach based on tasks. This formalism explicitly organizes knowledge into task-oriented contexts. As the name implies, CxG also provides a method to visually represent the knowledge using graphs in some ways similar to decision trees. The graphs offer users not familiar with knowledge-base system design with a straightforward means to directly review the knowledge (Brézillon, 2003b).

By modifying the Contextual Graphs representational paradigm to allow for explanations, a full and complete explanation can be provided in the context of a decision support system. This explanation gives insight into the reasoning process that the system uses to achieve its decision, as well as to provide further information to the user about how to accurately navigate the decision process.

Types of Explanations

As stated by Spieker (Spieker 1991) and more recently outlined by Roth-Berghofer (Roth-Berghofer 2004), there are five kinds of useful explanations.

- *Conceptual explanations:* The goal of this kind of explanation is to map unknown concepts to known ones. They are of the form "What is …?" or "What is the meaning of …?".
- *Why-explanations:* These explanations describe the cause or the justifications for the facts or the occurrence of an event. They are of the form, “Why am I performing this action?”
- *How-explanations:* They are a special case of Why-explanations, describing processes that lead to an event by providing a causal chain. They follow the form of “How did this action, or result, come to pass?” A How explanation follows the chain of events that led to the action. This is different from the Why explanation that gives only the immediate factors that contributed to the result.
- *Purpose-explanations:* This type of explanation describes the purpose of a fact or object. They are of the form, "What is …. for?" or "What is the purpose of …?"
- *Cognitive explanations:* Cognitive explanations explain or predict the behavior of intelligent systems on the basis of known goals, beliefs, constraints, and rationality assumptions. A Cognitive explanation will give the entire chain of events that the system reasoned.

The conceptual, why, and purpose explanations are static explanations. Expert systems answer such questions by using the knowledge contained in their (static) knowledge base. The how-explanation and cognitive-explanation are a description of the steps the system

has taken to develop the decision. These explanations are created dynamically depending on the path that the user takes through the system. This type of explanation illustrates to the user the reasoning process that the system uses to make decisions.

Requirements of an Explanation

Swartout and Moore (Swartout & Moore 1993) propose five aspects of a good explanation in the context of second generation expert system: Fidelity, Understandability, Sufficiency, Low Construction Overhead, and Efficiency.

- *Fidelity* means that the explanation must be an accurate representation of what the system does. Therefore, the explanations must be based on the same knowledge that the system uses for reasoning.
- *Understandability* is that the content of the explanation must be understandable to users. They should use terminology within the knowledge domain as well as allow for follow-up queries if the user requires further information.
- *Sufficiency* requires that the system has enough knowledge represented within its knowledge base in order to answer the user's question.
- *Low Construction Overhead* requires that the explanation must impose a light load on the development of the system. A load imposed on the construction of the system should be rewarded by easing the work load on another phase of the system's life-cycle.

- *Efficiency* deals with the response time that the system requires to make its decisions. The run-time efficiency of the system should not be hindered by the explanation feature.

These aspects of a good explanation were kept in mind when developing the explanation facility for the project as well as during the implementation phase. More specifically the aspects of fidelity, low construction overhead, and efficiency were taken into account when developing the explanation facility. When gathering the information to be incorporated into the system's knowledge base, the aspects of sufficiency and understandability were taken into account. The former aspects are those that are related to the design of the system, while the later aspects are those related to the inclusion of the actual knowledge that goes into the explanation.

Explanations in other Works

There has been a considerable amount of work in the field of explainable AI. Explainable AI began to come to the forefront in the late 1970s with attempts at explaining early rule-based systems behaviors. Since that time, there has been a steady progression of research in this research area. Today, most of the research centers on explaining the behavior of systems that teach users about a certain subject, such as military training for soldiers.

Early work in producing a viable explanation for an expert system came through the use of Rule-Based reasoning system. Rule-Based systems store their knowledge in a repository of facts and heuristics. They process the information contained within their knowledge base by firing rules using a chaining methodology, such as forward or backward chaining. These systems refer to their rules when giving an explanation of their

activities. A record of what rules were fired is kept by the system and then reported to the user as an explanation. MYCIN, a notable early system used for medical diagnosis, used this type of methodology to explain how it reached a diagnosis.

MYCIN used a complex system of rules to diagnose a patient and suggest treatment (Shortliffe 1976). It was realized early on that a doctor would not be willing to accept a diagnosis when they were not privy to the reasoning that went into the decision-making process. MYCIN was able to explain its reasoning at any point in a consultation by listing the rules it has under consideration at that moment. The problem is that a user may not always understand the rules or their purpose (Leake 1996). There is no guarantee that the regurgitated rules will provide an understandable explanation, nor is there any reason to suggest that the explanation would help the user make a subsequent decision. Furthermore, MYCIN's rules contained implicit knowledge related to the diagnosis that was not accessible to the explanation component of the system. Therefore, this knowledge was not available to the user accessing the system.

NEOMYCIN is also a medical diagnosing consultation system in which MYCIN's knowledge base is reorganized and extended for use in GUIDON, a teaching program (Clancy & Letsinger 1982). NEOMYCIN, and hence GUIDON, were developed to teach students medical diagnosing strategies. A set of meta-rules were constructed that would interpret the diagnostic rules. The drawback of this, besides the significant added complexity, is that the logic used to make the diagnosis is not the same logic used to compile the explanation. The user is still separated from the actual decision making process that the system used to reach the diagnosis.

An entire family of systems was developed from MYCIN, NEOMYCIN being one, and is shown in the following figure. Although the lessons learned from these systems were valuable to the explainable AI community, the system themselves were never used for their intended purpose.

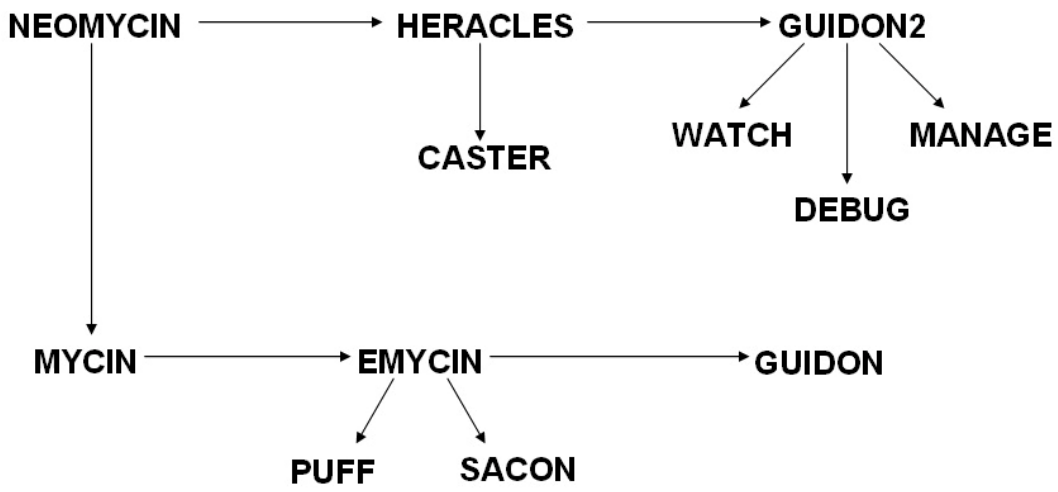

Figure 1: MYCIN Family of Systems

In the 1990s Case-Based Reasoning (CBR) was a very popular knowledge representation methodology. Case-Based systems use various algorithms to match a current situation to previous historical cases contained within its case library. The strength of CBR is that cases are an excellent way to present information to a user because cases are easily understood and often instantly recognizable as being pertinent to a given situation. Two common ways for deriving an explanation from a case-based system are the knowledge-intensive and knowledge-light approach (Cunningham, Doyle et al. 2003).

Explanations generated in a knowledge-light approach are simply comparisons to the previous case. When a user is asked to explain why the outputted case was presented,

the system would respond, “Your case is similar to Case A. In Case A this was the outcome.” This output can be useful in many situations; however, this approach gives the user no insight into the logic used to make the decision. The knowledge-intensive approach involves using a hybrid system consisting of case-based reasoning in conjunction with rule-based or model-based inferencing that can be used to generate explanations (Doyle, Tsymbal et al. 2003). The limitation of this approach, as in the case of NEOMYCIN, is the inference engine used to generate the explanation does not use the same algorithm that generated the decision. Therefore, the same logic used to generate the decision is not employed to reach the explanation. This provides a system that may be able to give an explanation of why a specific case was chosen; however, it does not explain how the inference engine of the system reached the decision.

The problem with explaining cases is that whether knowledge-light or knowledge-intensive, case-based explanation is *case-based* (Cunningham, Doyle et al. 2003). In other words, the explanation produced by a case-based system will be a similar case and not the logic used to select the case.

In the 1990s, through today, research began to develop using context when representing knowledge. Some notable contributions in this area came from Brézillon (1999), Carenini (1993), Gonzalez (1999), Karsenty (1995), and Sowa (1992). These researchers show that using context provides a way of narrowing the knowledge needed to produce a decision to only the factors which are relevant to the given situation. This allows the system to ask only questions relevant to the given situation and produce customized results to the user based upon this situation.

Brézillon (1994) describe using context to produce an explanation to the user. The ability of a system to produce explanations is dependent on the context in which the system and the user interact. Karsenty and Brézillon (1995) describe using user and system interaction to produce a viable explanation. Using the user to gather the context of the given situation, the system is then able to make a decision based upon that context. Because the system then knows the context the decision was made in, it is able to explain exactly what factors contributed to the decision.

Recently, many researchers are using context in generating explanations for their knowledge representation systems, even if not mentioned specifically. The Institute for Creative Technologies at the University of Southern California is currently working in the field of explainable AI in relation to designing military training aids. The have previously achieved success in the military and commercial market with their work in developing the video game *Full Spectrum Warrior* (Macedonea 2005), a game which lets the user act as squad leader in an urban combat situation. The current work places the user in simulated military exercises where their activities, and the activities of the artificially intelligent opponents, are recorded. The users are then able to query the system about the actions that the artificial soldiers took in given situations (Lane et al. 2005). This ability to question the system to explain its decisions is an important part of training. It allows the users to query individual entities, from a preset list of questions, about what they were doing in a given scenario. The entity will give a response to the preset question based on their status at that given point in time. Currently the ICT is working on expanding their XAI (eXplainable Artificial Intelligence) tool to be able to answer why type questions of the format, “Why did the unit pause in the middle of

executing a given task?" This indicates that they would need to take into consideration the relevant context that influenced the entities given situation.

To be comprehensive I would like to note the intelligent systems area of neural networks. Because of the way neural networks manage their knowledge, they are inherently unable to explain their decisions (Roth-Berghofer 2004). Neural networks can be looked upon as a black box containing a network of nodes connected by weighted links. The network is trained by presenting it with examples that contain the corresponding answers. Each time a new example is presented, the weights are adjusted in order to produce the correct output. Because a neural network solves a problem by activating links in accordance to their weights, neural networks are not able to easily give an explanation of its activities. Mathematical definitions may be produced, but a definition of this sort would not be an effective explanation for an average user who is not proficient in such areas. When asking the system "why" it produced an output, it would only be able to list the activated links. This does not constitute an explanation of the decisions that went into deriving the logic used to formulate the result. Unlike in CBR, the cases and generalizations used to train the neural network are not available at run time for comparison to the outputted result. The training cases are not part of the logic the system uses to make decisions; therefore, they are usually discarded after training.

The work produced to provide an accurate representation of how a system has reached a decision is still in progress. Currently, satisfactory explanations for intelligent systems do not exist. Explanations have been produced to explain the output of a system. However, either they are a regurgitation of the rules activated, which is often

unintelligible to the user, or they require a separate system to analyze the results in order for them to be understandable. The difficulty in explaining the decision making process can be attributed to the way in which the knowledge is represented within these systems. To produce an explanation that is effective and efficient, the knowledge must be represented in a way that will facilitate the process.

CHAPTER 2: PROBLEM DEFINITION

There is no generally accepted method to produce sufficient explanations in today's intelligent systems. As outlined previously, there have been many attempts to solve this problem. Some of these attempts have met with success in increasing the user's understanding of the system and thereby increase the use of the system by the user. Other attempts have shown flaws in the explainability of the knowledge representation paradigm on which they are implemented. Namely, they have shown that it is difficult to produce an explanation to the user that makes sense of the logic the system used to arrive at its decisions.

Because of the way Contextual Graphs represent and process knowledge, it is possible to produce an explanation of the system's actions at any point in the decision making process. The explanation produced could explain what the system is currently doing as well as the factors that have led to the system's current state. This permits an effective and efficient explanation of the system's decision making process.

As discussed previously (Schank 1986), providing an explanation that is both effective and efficient will increase a user's understanding of the system. This will, in turn, increase their trust in the system and consequently, increase the use of the system. There are two temporal formats proposed for the explanations: pre-explanations and post-explanations. A pre-explanation is an unprompted explanation designed to help the user accomplish a task or to help navigate the decision making process of the system. A post-explanation is given upon a request by the user to explain why the system provided a specific decision.

Hypothesis

It is asserted that the nature of Contextual Graphs facilitates the incorporation of explanations into an intelligent system that will increase the user's understanding of how the system functions. Since Contextual Graphs are capable of producing both the pre-explanation and the post-explanation, both will be implemented in the prototype system. It is further hypothesized that a pre-explanation will be more useful in helping the user navigate the decision making process. The post-explanation, on the other hand, will be more useful in providing an explanation regarding why the system made its decisions.

Contributions

The contributions of this thesis are listed below:

1. Enhance/Extend contextual graphs to include explanations. (The explanations for Contextual Graphs discussed are the first formalized attempt to provide an explanation facility to CxG. It is not proposed, however; that this is an official adaptation to the Contextual Graph paradigm. It is a legitimate attempt to investigate and possibly produce a valid explanation feature for CxG.)
2. Evaluate the preferability of pre-explanations and post-explanations by users of a decision support system.
3. An explanation facility added to a Prototype Decision Support System that implements Contextual Graphs that is able to express. pre-explanations and post-explanations to a user.
4. Test Data reflecting the pilot evaluation of the explanation facility of the system.

CHAPTER 3: EXPLAINABLE CxG CONCEPT

As stated in Chapter 1, the Contextual Graph (CxG) formalism offers a method of representing knowledge in a structured approach based on tasks. This formalism explicitly organizes knowledge into task-oriented contexts. As the name implies, CxG also provides a method to visually represent the knowledge using graphs in some ways similar to decision trees. The graphs offer users not familiar with knowledge-base system design with a straightforward means to directly review the knowledge (Brézillon, 2003b).

The Role of Context in CxGs

The role of context can greatly influence how knowledge is organized and utilized in a knowledge-based system. Pomerol and Brézillon (Pomerol &Brézillon, 2001) define context from an engineer's position “as the collection of relevant conditions and surrounding influences that make a situation unique and comprehensible”. This definition illustrates the potential value of context in providing focus in a system.

Context allows the system to focus on the factors that are important for the decision-making process in a certain situation. Factors that are not within the problem domain are not considered because they are irrelevant to the outcome. Contextual graphs allow for the separation of knowledge into that which is specific to the given context and that which is not. This allows for a quick and accurate decision to be made based upon the knowledge that is relevant to the given situation.

Components of CxG

The contextual graphs formalism is a knowledge representation paradigm specifically developed to take context into account in decision-making (Brézillon, 2003a). A

contextual graph is an acyclic, directed graph that consists of a single input and a single unique output. CxGs use various nodes connected by directed arcs to represent the knowledge. The contextual, recombination (C1, C2, etc.), action (a1, a2, etc.), and activity (A1, A2, etc.) nodes symbolize the various characteristics used in the decision-making process. Figure 1 is a contextual graph using the above mentioned components.

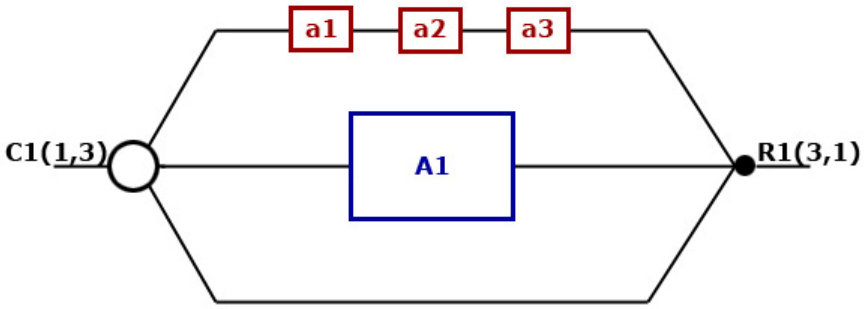

Figure 2: Contextual Graph with Activity

The careful arrangement of the nodes in a graph presents a functional model of the decision-making process (Brézillon, 2003b). The reasoning mechanism follows the direction of the graph and determines context at each contextual node. This evaluation of context decides which of the divergent paths to pursue. The path of all the nodes visited yields a specific practice that also can be used to trace the logic of the system. A more detailed explanation of contextual graphs can be found in Brézillon, (2003a).

The components of Contextual Graphs are divided up into individual nodes. This allows for reuse of individual nodes through out the graph.

Contextual Nodes

A Contextual Element is represented by a pair of nodes, a Contextual node and a Recombination node. The contextual node is represented as C(1,n) and its corresponding

recombination node is represented as R(n,1), where n represents the number of exclusive branches for said node. In Figure 2, the contextual node shown would be shown as C(1,3) and its corresponding recombination node would be R(3,1). As shown, a contextual node has a single input but may have several diverging branches, each of which represents the different contexts that can be determined by the node. All branches from a contextual node eventually converge back to the same recombination node when the context no longer plays an active role.

When progressing through a contextual graph, the contextual nodes play the role of creating the specific context that is relevant to the given situation. Once a recombination node is reached, the context becomes de-instantiated and no longer plays a role in the decision making of the system.

Actions & Activities

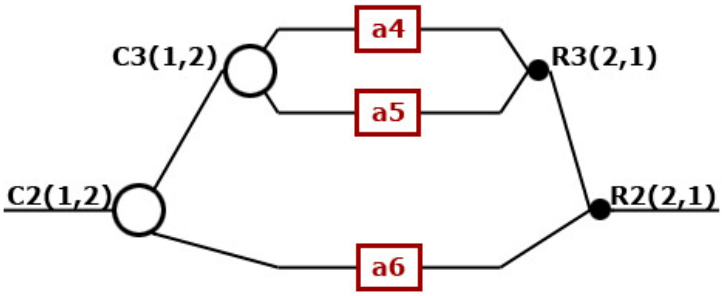

Figure 3: Expanded Activity from Figure 1

Action nodes are executable methods. Activity nodes are considered complex actions that are usually repeated within a contextual graph. Activities are a directed graph with a

unique input and a unique output. They may be composed of multiple actions and/or a complete contextual graph with its own corresponding actions and activities.

Context Representation in CxG

The identification of the role of context results from the distinction between procedures and practices. A procedure is an officially defined process to solve a problem, and a practice is the actual process followed by an agent (human or automated) to solve that problem under a specific context. This is also the basis of how CxG incorporates new knowledge - a technique called *incremental knowledge acquisition.* This is described in the subsequent sub-section.

CxG divides context into three different categories: *contextual knowledge, external knowledge,* and *proceduralized context.* Contextual knowledge represents knowledge that is used to constrain the decision-making process. All other knowledge at the beginning of the task is external knowledge and it is considered irrelevant to the problem. The contextual knowledge is represented by contextual nodes. When the contextual knowledge is instantiated, it transforms into a proceduralized context. This proceduralized context eventually becomes de-proceduralized back into contextual knowledge.

Further consideration on the conversion between contextual knowledge and proceduralized context illustrates how CxG uses context. Each branch of a contextual element (a contextual and recombination node pair with the corresponding links connecting them) represents the possible alternative contexts for that element. The decision made at the contextual node determines the context that constrains the situation.

Once the context is determined, the contextual knowledge now becomes a proceduralized context. The proceduralized context constrains the decision-making process by allowing only the subsequent actions, activities and contextual elements on its branch to be executed. The proceduralized context then reverts to contextual knowledge once the path has reached the recombination node of the contextual element. This process illustrates the point when the context no longer has a constraining effect on the decision-making process (Brézillon, 2003c).

When using procedural context, it is important to note the path taken out of each contextual node. Referring to Contextual node C1 in Figure 4, we can observe that there are two paths proceeding from the node. When using procedural knowledge, the path taken is referred to by the following designation: C1.1 for the first path proceeding from the node and C1.2 for the second path proceeding from the node. Likewise, the path taken out of contextual node C2 can be shown as C2.1 for the first path proceeding from the node and C2.2 for the second path proceeding from the node. Representing the contextual nodes in this way provides a quick and efficient way of refering to the specific context instantiated when proceeding from a contextual node.

Figure 4 and Table 1 are, respectively, an illustration of the contextual knowledge and proceduralized context when stepping though a contextual graph.

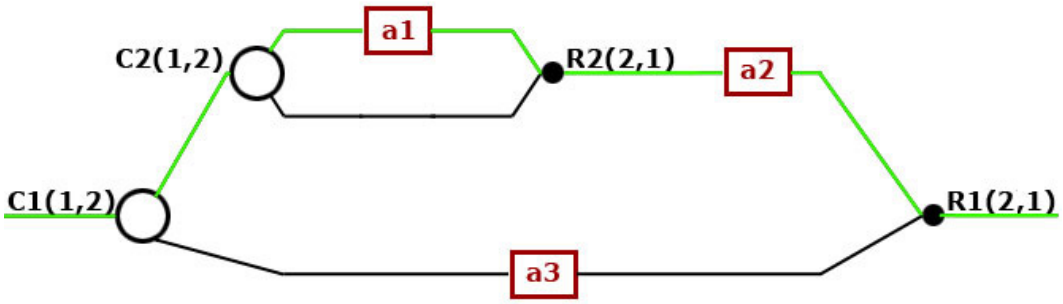

Figure 4: Procedural Context Example

Table 1: Proceduralized Context Example

Step	Context From	Contextual Knowledge	Procedural Context
1	C1	{C1, C2}	Ø
2	C2	{C2}	{C1.1}
3	a1	Ø	{C1.2, C2.1}
4	R2	{C2}	{C1.1}
5	a2	{C2}	{C1.1}
6	R1	{C1, C2}	Ø

Knowledge Acquisition in CxG

A contextual graph is initially developed to represent established procedures. In reality, an agent adapts a procedure to accommodate the contexts of the situation to form a practice (Brézillon, 2003b). A practice can be considered the overall path that had been taken throughout the situation. The number of practices cannot be determined when the initial graph is built. The technique of incremental knowledge acquisition introduces a method of incorporating new practices that had not been previously represented (Brézillon, 2003a).

This technique is built upon the interaction between the user and the system. Once the user has solved a problem, he/she can report the sequence of actions to the system. A

discrepancy in the sequences of actions between the user and the system can be viewed as possibly a new practice. The logic between the system and the user is then compared to determine whether any new contextual elements are needed to incorporate the new practice. This integration of new knowledge supports the characteristic of CxG that it may evolve to expand its knowledge base (Brézillon, 2003a). Likewise, this technique assists in building tasks where originally insufficient knowledge had been included.

To illustrate the point, as one progresses along the graph in Figure 4 and arrives at the contextual node C2, two choices are offered. If at this point, neither of the choices is correct for the given situation, a new choice may be added to the contextual node, see Figure 5. This then creates a new path in the graph with the corresponding action or activity, in this case a4, being added. This allows for contextual graphs to adapt to new situation and facilitates the addition of new knowledge. Currently, this information needs to be added by the knowledge engineer and is not compiled into the graphs dynamically by the system.

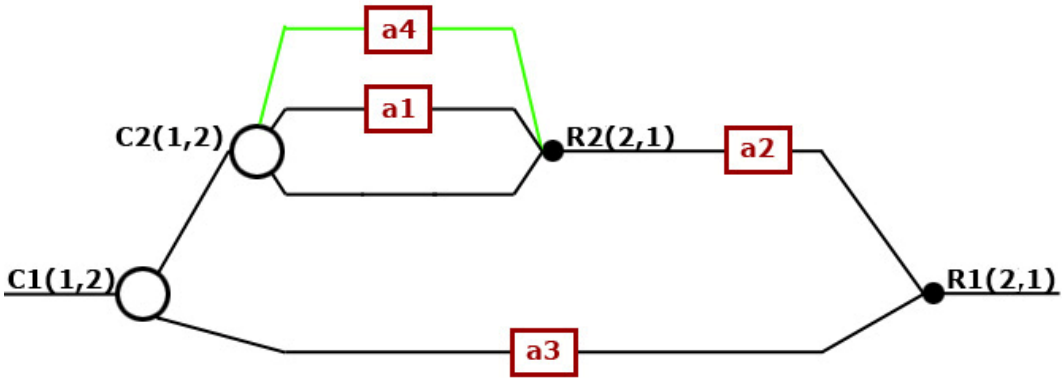

Figure 5: Knowledge Acquisition

Explanations in Contextual Graphs

Preserving the integrity and the internal workings of Contextual Graphs was a major objective when developing a paradigm for representing explanations in Contextual Graphs. To this effect, the following points were observed: The Explanation facility must not change the components of Contextual Graphs, nor should it affect the logic of Contextual Graphs.

The explanations proposed are based on the types outlined in the previous chapter: Purpose, Why, How, and Cognitive. The Conceptual-explanation was not implemented within the system. Conceptual-explanations are the questions of the type "What is..?" or "What is the meaning of…?" The knowledge contained in Contextual Graphs is of known procedures and concepts. Therefore, questions that attempt to map unknown concepts to known concepts are not applicable. In the case of a cup of coffee, one is more likely to ask, "How do I make this cup of coffee?" instead of "What is coffee?" Stating what coffee is does not give a reasonable user any deeper insight. The goal of adding explanations to contextual graphs is to give the user a deeper understanding of the system's reasoning process in order to increase trust in the system and thereby increase its use. Conceptual explanations are treated as rare exceptions and are better left to dictionary queries.

It must also be understood that there are two temporal formats to the explanations proposed, a pre-explanation and a post-explanation. The pre-explanation is a guide to help the user make a decision, or answer a question posed by the system. The post-explanation is a description of how the system made a decision based upon the user input.

Pre-explanations are those that help a user progress through the Contextual Graph. They are a run-time explanation produced to the user Pre user input. Purpose-explanations are considered instances of Pre-explanations. They help the user further understand the Action or Activity that the system wants the user to accomplish. Why explanations are also of the pre-explanation type. They are implemented at a contextual node and help the user make their decision on which path to proceed.

Post-explanations give a deeper look into how the system has reached a certain decision. This type of explanation is produced for the user after they have given the system input. They show the user why they have traversed to a certain point within the graph. How and Cognitive explanations are examples of Post-explanations.

Explanations Types

The following section will seek to explain the different types of explanations the system produces. The pre-explanations, purpose and why, are both associated with specific node types within contextual graphs. The post-explanations, how and cognitive, are based upon the path taken through the graph.

Purpose Explanations

Actions are designed to be reusable elements within Contextual Graphs. One may do the same action in multiple contexts and have a different reason for doing so each time. The Why of doing the action is based within the contextual element in which the action occurs. An example of this scenario could be going to the grocery store. One could go to

the store to purchase a loaf of bread if hungry. One could go to the store to purchase aspirin, if one has a headache. One could go to the store to pick up a paycheck, if they were employed there. In each instance, the participant went to the store but the explanation of why they were there depended on the context of their individual situations.

Purpose explanations are pre-explanations designed to help the user execute an action in the contextual graph. "What purpose does this action serve?" or "What is this action for?" these are the types of questions that may be asked about a specific action. This allows for the incorporation of knowledge that helps the user accomplish the task, allowing helpful advice about how to complete the action that is presented, and allow for the incorporation of first hand experience in completing said action. Furthermore, importantly, the purpose-explanation is not specific to the context where the action is presented.

By not making the purpose-explanation specific to the context where the action appears, the action is able to be reused at any point within the graph. This re-use of actions within a graph preserves the structure, and one of the advantages, of Contextual Graphs. The action and its explanation is the same, regardless of where in the graph it is situated:

Purpose-Explanation (action) = {Explanation [action]}

Why Explanations

The Contextual Node is where the logic of the graph resides, where the decisions are made. As stated previously, the contextual node is essentially a question that when answered places the user on the correct path in the graph. The why-explanation helps the

user make a decision by explaining the logic that went into asking the question. This allows for the incorporation of knowledge related to why the user is asked to choose between the contexts presented as well as being able to present factors to be considered in making the decision. It should be noted that the last point of presenting factors to be considered in the decision making process can be represented in the original paradigm of Contextual Graphs. By dividing the factors up into their own contextual nodes, one may ask the user multiple questions to determine exactly which path to direct them to. Representing the factors in this manner can introduce many simple questions into the system which can lead to annoyance by the user. A why-explanation can be represented in the following manner:

Why-Explanation. (contextual node) = {Explanation[contextual node]}

Taking an example from the AlexDSS prototype, discussed in the proceeding chapters, we will look at a scenario where a user encounters a contextual node in the contextual graph *Meeting Agenda.* The user is asked, "How many projects are to be presented at the meeting?" The corresponding why-explanation is given as, "The meeting format will change based on how many projects are selected."

How Explanations

Where the purpose-explanations and why-explanations are instances of pre-explanations, how-explanations are of the post-explanation type. After the user has progressed through the graph and is presented with an action, the how-explanation is able to display the factors that went into the system's decision-making process. Essentially, the how-explanation describes how the user reached a certain point within the contextual graph.

It is important to distinguish the how-explanation from the why-explanation. The why-explanation asks, “Why am I being asked this question?” or “Why am I doing this action?” The explanation provided is the immediate factors that went into why that specific node in the graph is being executed. The how-explanation is the procedural chain of events that lead up to why an action was reached. This includes all the factors that went into the execution of the action node. Essentially, the how-explanation is a why-explanation with a longer memory.

Brezillon describes using the proceduralized context to give an overall explanation about how one reached a certain instantiation in contextual graph (Brezillon 2003a). This proceduralized context is the basis for the how-explanation. If one looks at the proceduralized context when a contextual node is reached, the proceduralized context will show all the contexts that where instantiated in order to reach that contextual node.

Each path proceeding from a contextual node is associated with its own explanation. This can be as simple as stating, “path 2 out of contextual node 1 was taken,” or it can be an in depth explanation describing the factors that went into deciding to take the chosen path. The how-explanation gives the entire context that is relevant to the current action. The how-explanation can be represented in the following manner:

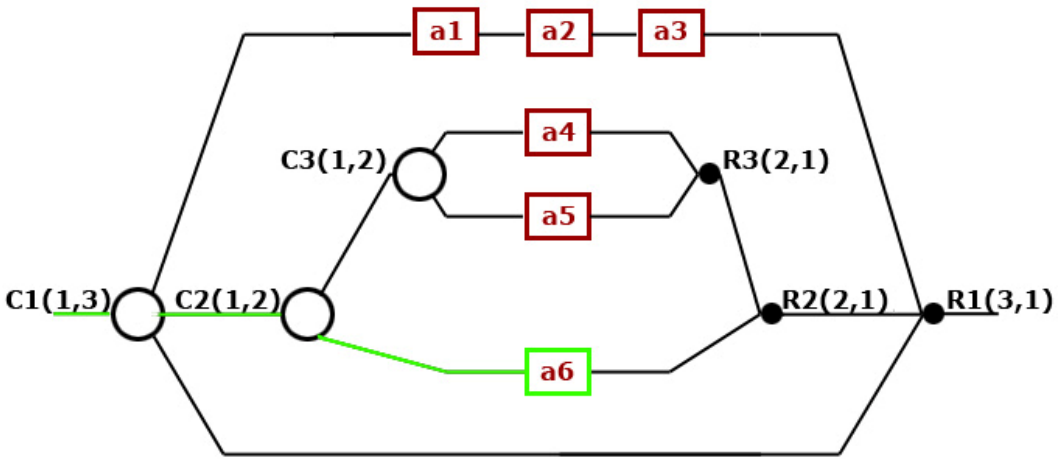

Figure 6: How Explanation

How-Explanation. (a6) = {Explanation[C2.2] + Explanation[C1.2**]}*

**C2.2 indicates that the second path was taken out of contextual node C2.*
***C1.2 indicates that the second path was taken out of contextual node C1.*

In this example, the justification for executing action a6 is given by the contexts that were instantiated in order to reach node a6. First contextual node C1 is reached. The user narrows the context of the situation by taking the middle path (C1.2), thus discarding the options that are associated with the alternative choices. Next the user chooses the second choice for contextual node C2 (C2.2), further narrowing the context of the situation. Thus, when action a6 is executed, the system is able to point to the context instantiations of C1.2 and C2.2 as those responsible for the execution. Because the how-explanation uses procedural context as its basis, the contextual node C3 is excluded from the explanation. C3 was never reached when traversing the graph and therefore it is not a contributing factor in the execution of node a6.

Cognitive Explanations

Proceduralized context works well for showing how one got to a certain node within the graph, but consider the following situation. When traversing through the graph, contexts are constantly being added and subtracted from the proceduralized context. When a contextual node is reached, that node is added to the proceduralized context; however, when the corresponding recombination node is reached, that instance of the contextual node is removed from the proceduralized context. This allows for a problem in the case of complex graphs where there are many paths to take to reach a certain point. When one reaches a recombination node, all the decisions that took place between the paired contextual and recombination node are lost. For a cognitive-explanation, the path that was taken to traverse the graph must persist. A final overview of the output of the system can be obtained by explaining the reasons for each action presented in the path taken through the graph. This can be accomplished by stringing together the how-explanations of each action taken. The reason for each action taken is based upon the context where the action appears. The following examples illustrate the cognitive-explanation that would be presented for the specific paths through the graph.

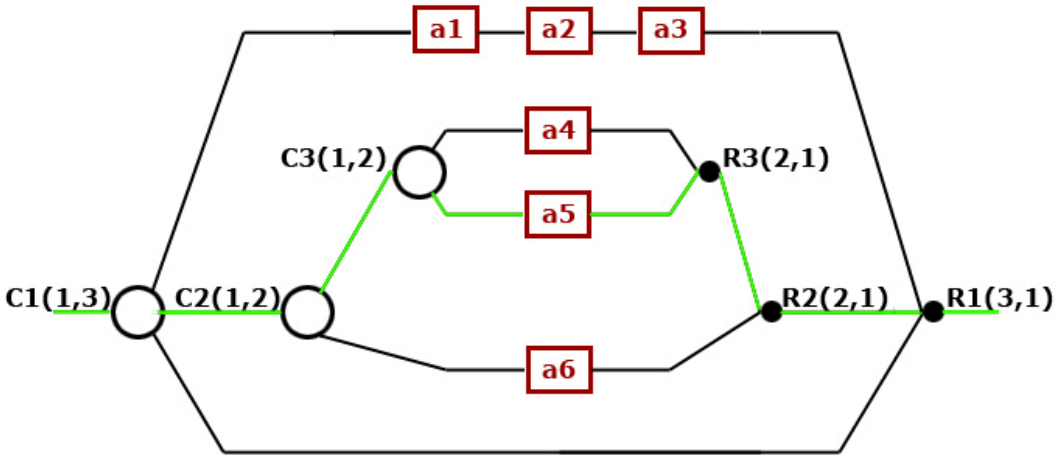

Figure 7: Cognitive Explanation Example 1

Cognitive-Explanation = {How-Explanation.(a5)}

Cognitive-Explanation = {Explanation[C3.2] +Explanation[C2.1] + Explanation[C1.2]}

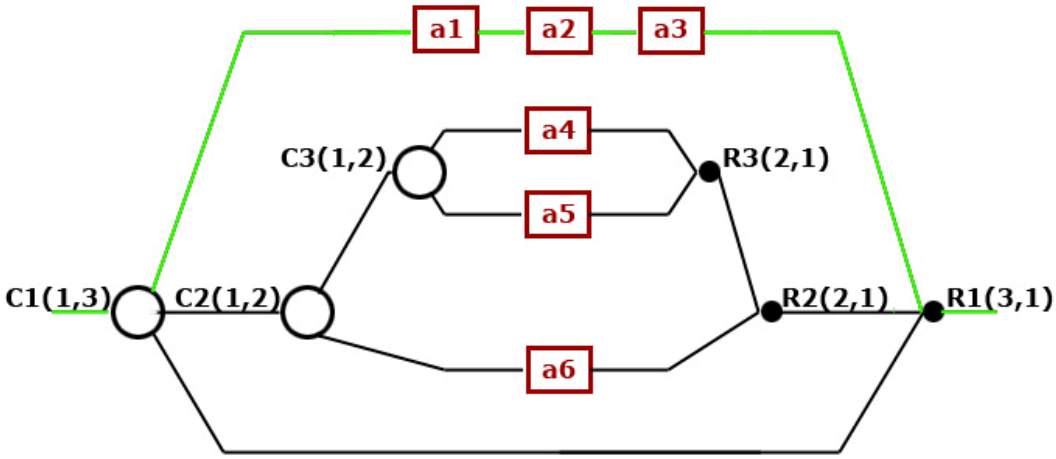

Figure 8: Cognitive Explanation Example 2

Cognitive-Explanation = {How-Explanation(a1)+How-Explanation(a2)+How-Explanation(a3)}

Cognitive-Explanation = {Explanation[C1.1] +Explanation[C1.1] + Explanation[C1.1]}

The following example is used to describe the instance of “no action” being taken. Since this research is concerned with producing explanations, some action must be taken in order to have something to explain, even if that action is “no action”. The idea of a null node is introduced to facilitate the cognitive-explanation in this case. This is demonstrated in Figure 9.

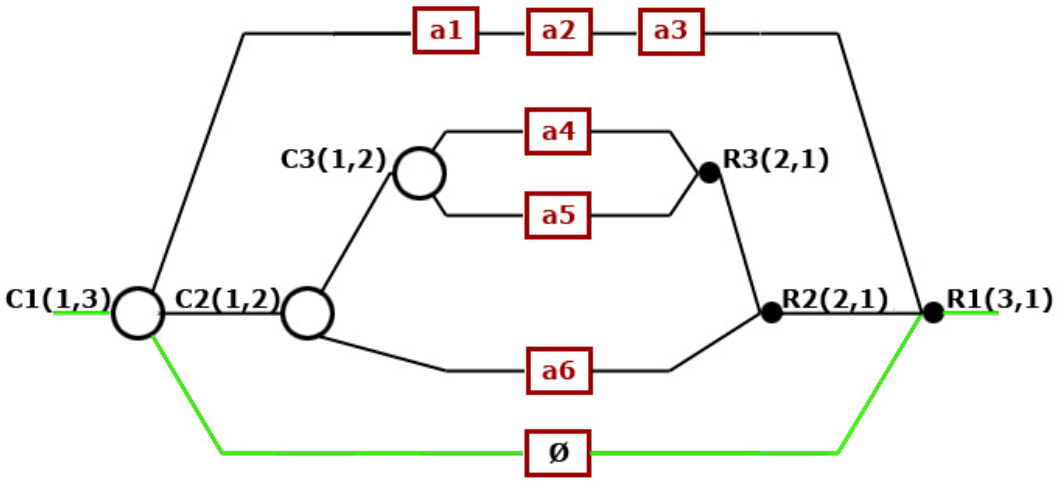

Figure 9: Cognitive-Explanation Example 3

Cognitive-Explanation = {How-Explanation.(Ø)}

Cognitive-Explanation = {E[C1.3]}

As stated previously, a cognitive-explanation is a dynamic explanation that is compiled while the user progresses through the graph. Every time an action node is reached, the how-explanation for that node is compiled. These how-explanations are then assembled together in sequential order. The cognitive-explanation is the assembled, sequential, how-explanations.

Summary

Explanations are presented in two temporal formats, pre-explanations and post-explanations. The pre-explanation is unprompted and is presented each time a contextual node or action node is encountered. Pre-explanations help the user navigate the contextual graph and provide advice on the best decision to be made in the user's current situation. The post-explanations are presented upon the user's request, and they provide the understanding of how the system progressed through the graph as well as the cause for why an action was presented. This allows for transparency in the system and is intended to give the user an idea of how the system reached an output.

The purpose and why-explanations are both pre-explanations are presented unprompted to the user. They are each imbedded into nodes within the contextual graph; action nodes and contextual nodes respectively. The how-explanations are contained within the individual paths proceeding from the contextual nodes. They are given by presenting the explanation for each context chosen, in other words, explaining each instantiated context within the procedural context. The cognitive-explanation is a final explanation presented when the user is done progressing through the graph. It is a post-explanation and therefore it is presented only upon prompting from the user. Cognitive-explanations are the sum of the how-explanations for each action taken within the graph.

CHAPTER 4: IMPLEMENTATION

A Decision Support System (DSS) using the Contextual Graph paradigm is being developed by the Intelligent Systems Lab at the University of Central Florida, in order to replicate the knowledge and decision making abilities of a long time Program Manager at the US National Science Foundation (NSF). The system will provide decision support in managing a major NSF research program. The prospective users for the system are the individual directors of each research center funded by the program, as well as the NSF Program Manager. This author participated in the initial construction of the Prototype with the ISL rest of the ISL team. The explanation functionality of the system was added later by the author who is solely responsible for its creation.

This author has spent the last year stationed at the National Science Foundation of the United States in Washington DC. The author was tasked with shadowing the NSF Program Manager whose knowledge is to be incorporated into the DSS. Observation of the PM was the main method used to gather the needed knowledge. Active questioning about specific topics related to the research program was used to supplement the observations made. The author accompanied the PM to the research meetings held around the U.S., and as such was able to observe the decision making process used by the PM when managing the program's research centers.

During the knowledge gathering process, a variety of knowledge representation systems were researched to determine which would be the optimal solution for representing this type of human problem solving knowledge. The systems researched included: Case-Based Reasoning, Rule-Based Reasoning, Context Based Reasoning, CommonKADS development methodologies and various hybrid combinations were also

looked into. Contextual Graphs was decided upon as a way to represent the knowledge in a form that was convenient to implement and manage. This system will be the first real world implementation of Contextual Graphs as a decision support tool. The system has been named AlexDSS; this is based on the first name of the program manager it is designed to emulate.

The development implementation of CxGs follows the design formalized by Brezillon (2004). Java was chosen as the system development language. The universal portability of Java and compatibility with on-going work by Brezillon and his group were compelling reasons for this choice.

Initial AlexDSS Prototype

Currently, the ISL team has built a prototype version of AlexDSS. The prototype was designed to test the implementation of contextual graphs for correctness and usability in a decision support system. Rather than implement all the domain knowledge of the subject, the prototype concerns itself with implementing an activity which contains all the contextual graph elements. The knowledge contained within the graph is concerned with assisting the user in planning a center meeting.

When discussing the idea of the system with the perspective users, the issue of trust was a constantly re-occurring theme. The users did not believe that the system would be able to correctly replicate the knowledge, or more precisely, the reasoning of the expert it was supposed to emulate. It was out of these discussions that the idea of implementing an explainable system developed. As shown in Chapter 1, explanations are the most effective way to increase the user's understanding of the system.

Knowledge Representation

The AlexDSS prototype employs all of the CxG nodes that had been presented earlier. The activity node is heavily utilized as a means to organize the knowledge into activities. Every type of node is a member of exactly one activity. Activities, on the other hand, can be members of many other activities or none. An activity node that doesn't belong to any other activity is considered to be a main activity. This type of grouping using activities adheres to the CxG formalism. Main activities may contain many nested sub-activities, which eventually create a hierarchal sequence of activities that maintain the directed acyclic nature of CxG (Brézillon, (2003a).

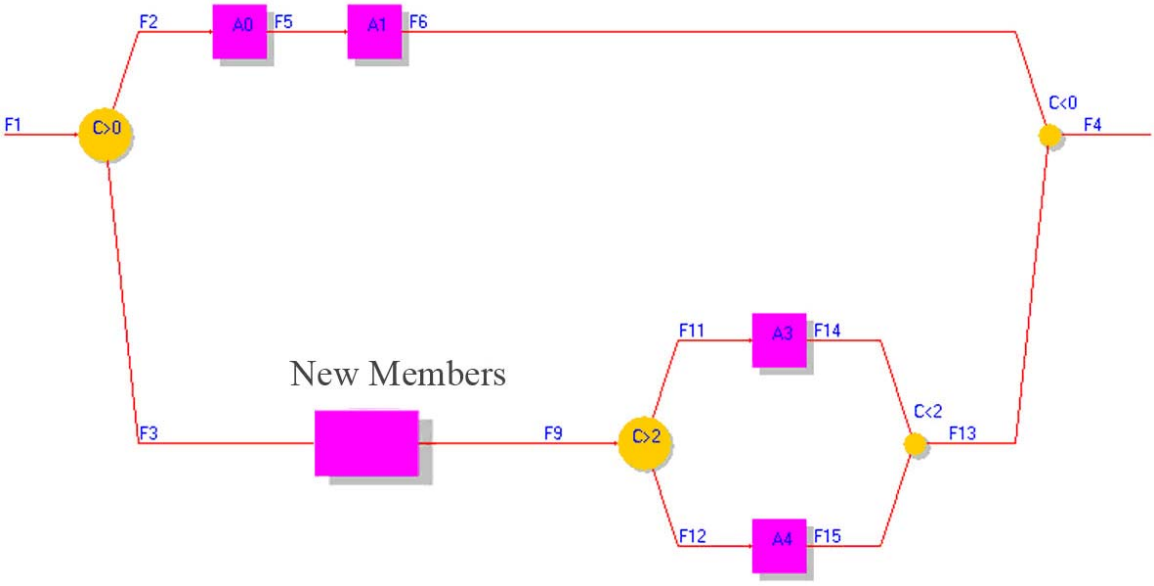

Figure 10: Type of Meeting

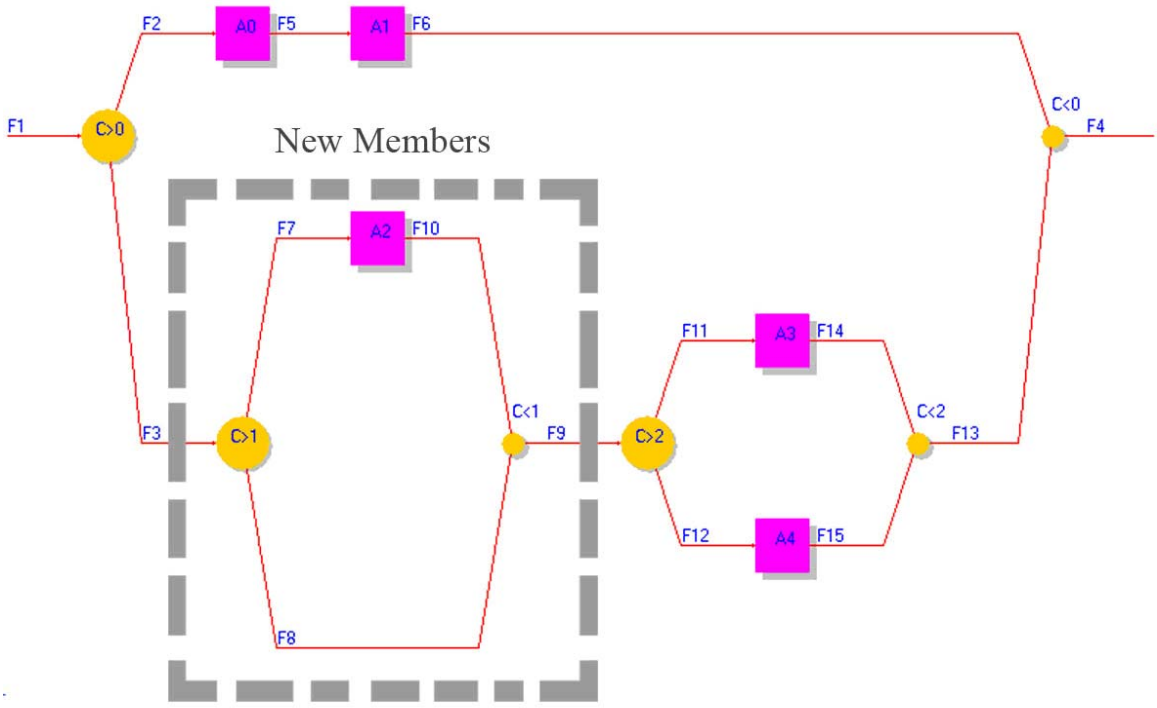

Figure 11: Type of Meeting w/ New Members Expanded

An example of this hierarchy of nested activities is illustrated in Figure 10 and Figure 11. This activity is named *Type of Meeting* and contains a sub-activity called *New Members*. The activity *New Members* is a member of the activity *Type of Meeting*. Likewise, *Type of Meeting* is a member of the main activity *Meeting Agenda* which is not illustrated here. Figure 9 depicts the *New Members* sub-activity as a node in the *Type of Meeting* activity. Figure 10 expands the *New Members* activity to illustrate the sub-graph that the activity represents.

The use of activities to group modules of tasks together offers many benefits. First the logical grouping immediately gives the sense of context to all of the members in the activity. The grouping of activities allows the developer to build graphs in several components. This allows the developer to build individual activities, whereupon they are

able to group them together to form larger activities or graphs. Implementing contextual graphs using various levels of abstraction promotes macro-management of the knowledge.

System Step-Through

The AlexDSS prototype utilizes CxG models and user interaction for the decision-making process. The AlexDSS prototype's engine starts with the main activity *Meeting Agenda*, and then navigates through the graphs contained within it. When a contextual node is reached, the engine uses a graphical user interface (GUI) to solicit information from the user. The GUI then sends the engine the user's reply which the engine uses to process the next step. The system employs this interaction with the user to determine the context of the problem.

When an activity is introduced, traversal through that graph halts at the activity node. Traversal then takes place inside the activity until that activity is complete, after which time traversal in the parent graph is resumed. To effectively manage nested activities in a computer system, a LIFO (Last In First Out) stack is used. This process is similar to proceduralization and de-proceduralization of context that Brézillon describes (Brézillon 2003a). Implementing a LIFO stack ensures that traversal is only occurring in one activity at a time.

The typical output of the AlexDSS prototype would consist of a practice to be followed. This practice does not always have a temporal relationship with the actions that build the practice. In this prototype, a typical action would have been “Reserve 15 min after lunch for the guest speaker's presentation”. Such an output presented to the user

immediately, when that action node is executed, may not have an intuitive bearing with the other actions. To effectively output a meaningful practice, the path of the main activity traversal is maintained. This path retains the action produced as well as all the proceduralized contexts and the activities that had been executed. The path is specific to the main activity which would then present the practice to the user once the main activity had been accomplished.

The output produced by the activity *Meeting Agenda* may not be typical of the completed AlexDSS. In the case of the activity *Meeting Agenda*, the goal is to output an entire agenda for a center meeting upon completion. In the case where a user is interested only in a specific output or answer, an action node may be presented immediately and not upon traversal of the entire graph. This is useful in the case where a user has a specific question about assembling a meeting agenda and does not have a need to traverse the entire graph.

System Organization

This prototype, similar to most modern knowledge-based systems, separates the knowledge and the reasoning mechanism (Gonzalez, 1993). The architecture of the system has been divided into three separate pieces: Knowledge Base, Inference Engine, and User Interface. Separation of the inference engine in the system allows reuse with any knowledge that is represented in contextual graph form. Figure 12 illustrates the relationship of the system components.

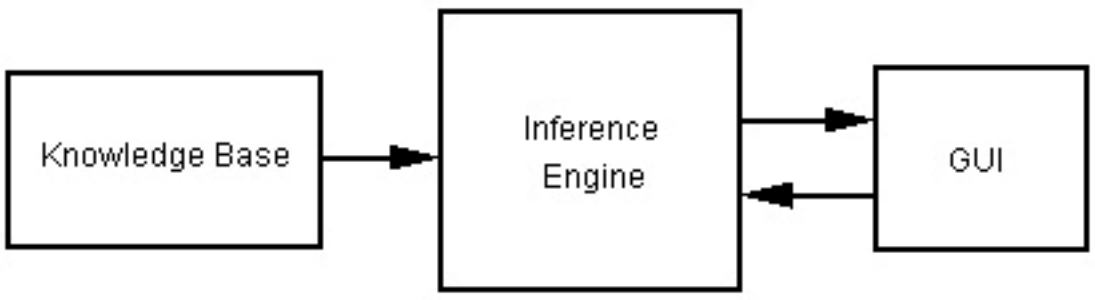

Figure 12: System Diagram

Knowledge Base

The knowledge the system contains is represented in the form of XML files. An XML file may contain a single activity or multiple activities contained within a larger activity. Using an XML format for the knowledge provides for an easy way to manage the knowledge. Implementing the knowledge base in this generic manner also allows for the reuse of the system for any type of knowledge, as long as it can be implemented in contextual graph form.

Originally, the knowledge base was composed only of the knowledge needed to construct the contextual graph. The addition of explanations was added to the contextual graph following the same XML format as the original file. <Explanation> tags where added to the contextual graph components discussed in Chapter 3. Subsequently, the XML Parser the system used to read the XML files was reprogrammed to reflect the change in knowledge representation.

The following is an example of a simple activity containing a contextual node and multiple sequential action nodes.

```
<Activity name="Duration of Meeting" beginid="0" endid="1">
  <Node id="0" type="Contextual" name="Duration of IAB meeting">
    <Context type="mapped"></Context>
    <Question>What is the planned duration of the meeting? (days)</Question>
    <Explanation>NSF recommends that you take 1.5 days for the meeting.
     &lt;br>Things to consider are where your companies are traveling from; if
     they are mostly from out of state, they are going to prefer a couple day
     meeting. Local companies will want a one day meeting.</Explanation>
    <Downstream id = "3">1.5</Downstream>
    <Explanation></Explanation>
    <Downstream id= "2">1</Downstream>
    <Explanation></Explanation>
  </Node>
  <Node id="1" type = "Recombination" name="Duration of IAB meeting">
    <Downstream id="null"></Downstream>
  </Node>
  <Node id="2" type ="Action" name="1 Day Meeting">
    <Action display="yes">A 1 day meeting is not recommended</Action>
    <Downstream id="1"></Downstream>
    <Explanation>It is very hard to get the feedback required from the companies
     in such a short time period. Presenting projects will take most of the day
     which leaves no time for LIFE form feedback.</Explanation>
  </Node>
  <Node id="3" type="Action" name="1.5 Day Meeting">
    <Action display="no">Have the IAB meeting on second day</Action>
    <Downstream id="1"></Downstream>
    <Explanation>You want to give the IAB time to speak with the researchers
     and amongst themselves about the projects before voting.</Explanation>
  </Node>
</Activity>
```

Inference Engine

An object-oriented architecture provides a suitable approach that complements CxG's modular nature. One can think of each node in the contextual graph as an object with the paths connecting the nodes as pointers to other objects.

The reasoning engine is divided into three different packages for reuse and manageability. First is the graphElements package, which contains all the various CxG nodes used in the system. Next is the xmlParser package, which contains the parser for the XML file containing the knowledge for the system in the form of Contextual Graphs.

Lastly, there is the cxgInference package that contains the various components of the reasoning engine.

The nodes in the prototype are represented by specialized classes that inherit from an abstract node class. This allows the reasoning engine to regard the nodes similarly during traversal. Each node contains a reference to the downstream node that is connected to it, in the contextual graph. The exception to this is for contextual nodes, which contain a list of downstream nodes for its branches instead of a single downstream node.

Each class in the graphElements package was altered in order to include the explanations needed. Each node had an explanation String programmed into the class with the corresponding function to get() or set() the explanation. For the purpose of implementation, the purpose-explanation and the why-explanation behave in the same manner. Both explanations are pre-explanations and are thus presented at the same time as the reasoning information contained within the node. Thus the implementation of the explanation is the same for the actionNode and the contextualNode. The contextualNode also contains an individual explanation assigned to each path proceeding from the node. These explanations are stored in a String array that is linked to the array containing the paths. The following Figures (13, 14, 15, 16, 17, 18, and 19) illustrate the class inheritance of the graphElements package, as well as the individual attributes of each class.

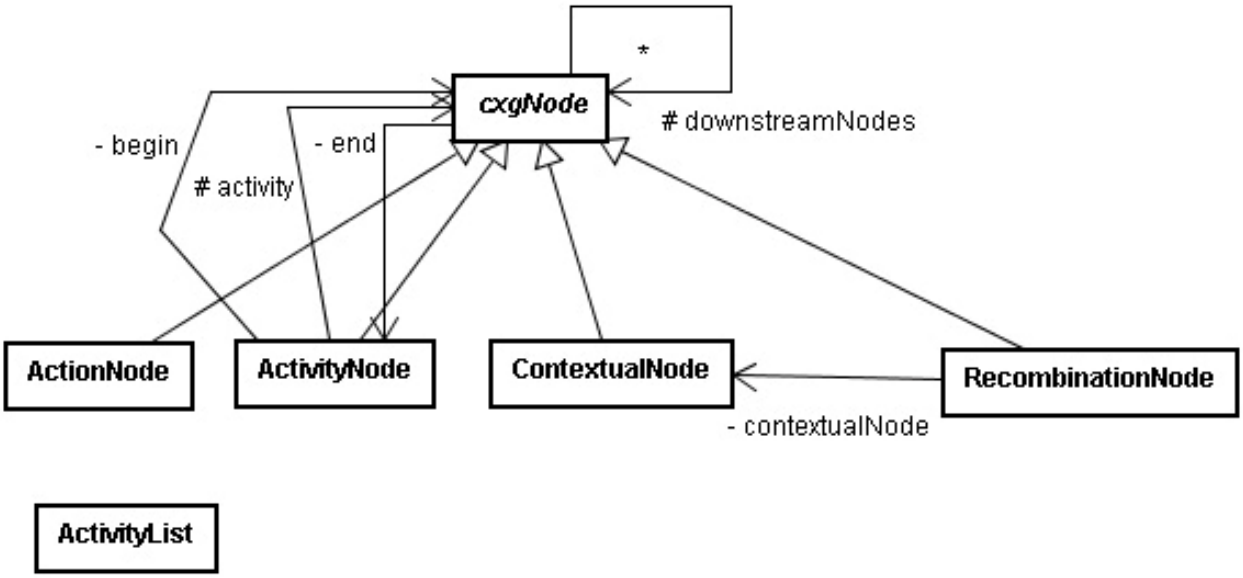

Figure 13: Inheritance Diagram of graphElements Package

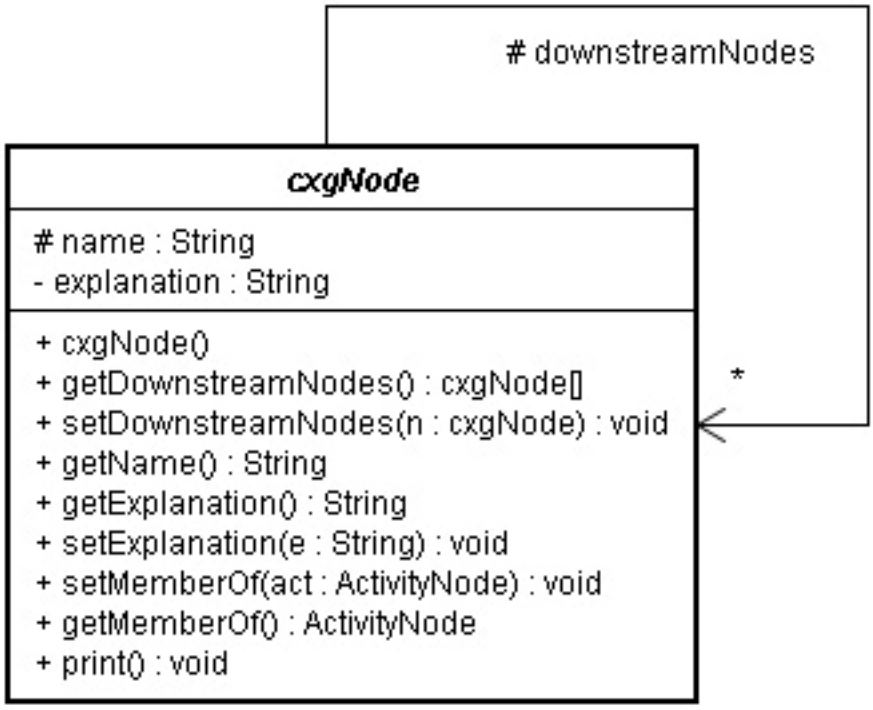

Figure 14: cxgNode Class Diagram

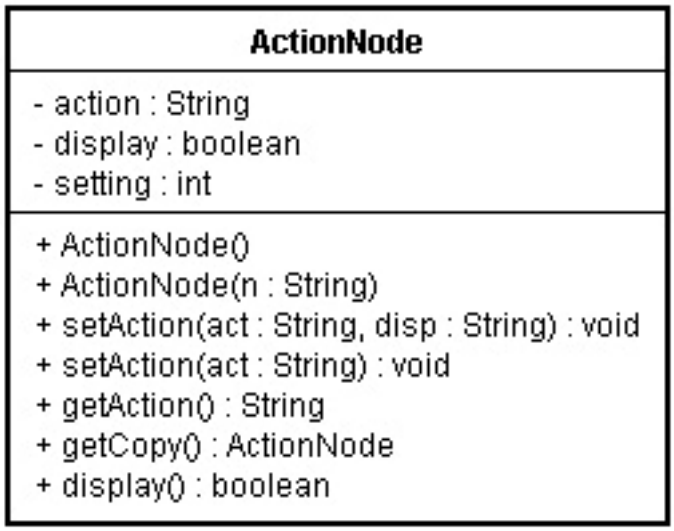

Figure 15: ActionNode Class Diagram

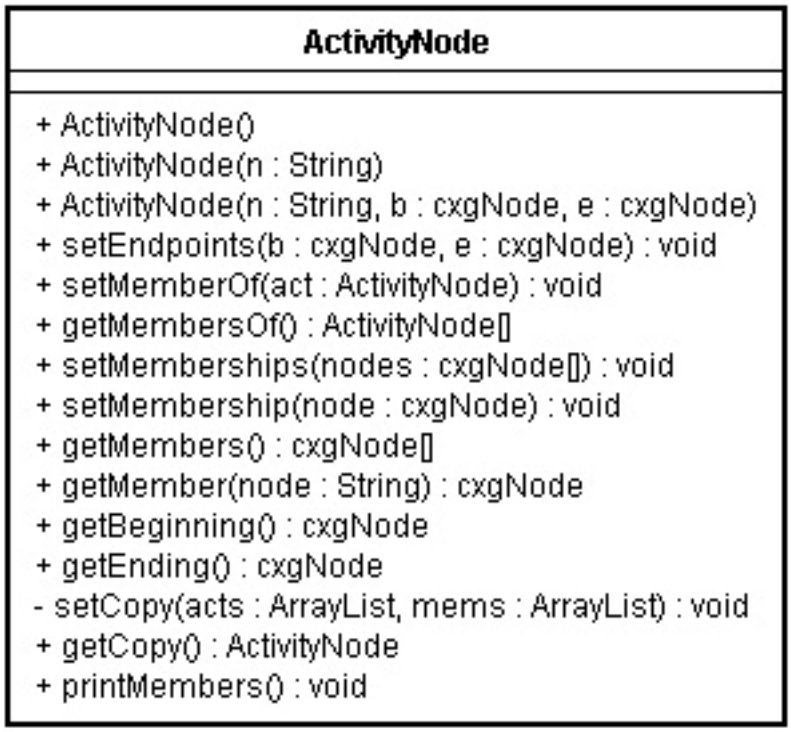

Figure 16: ActivityNode Class Diagram

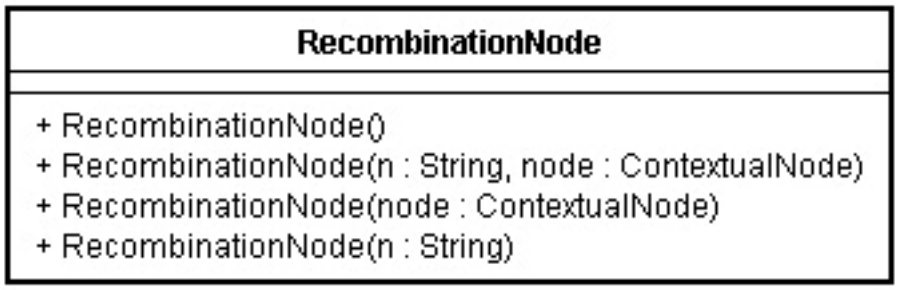

Figure 17: RecombinationNode Class Diagram

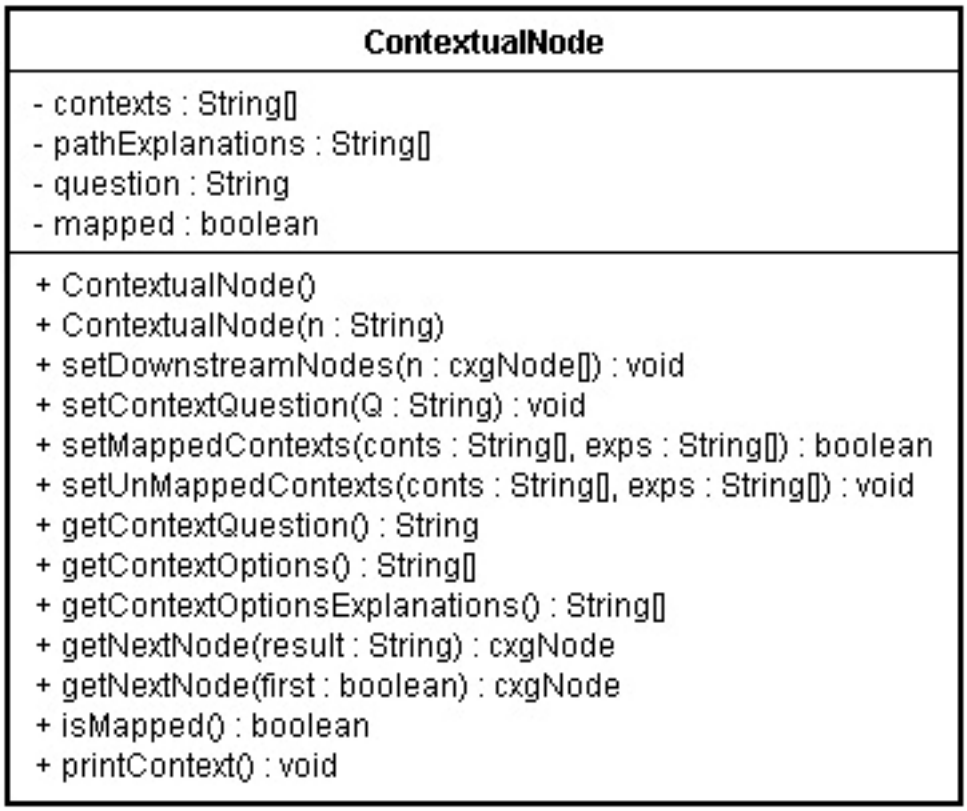

Figure 18: ContextualNode Class Diagram

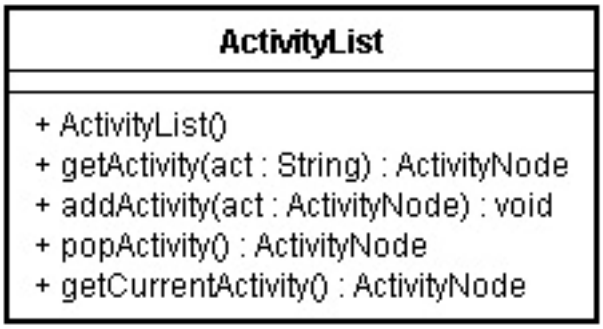

Figure 19: ActivityList Class Diagram

The xmlParser package contains a parser which, upon start-up, dynamically builds all the graphs stored in the knowledge base. The parser processes each XML file and links all the corresponding activities together to form the contextual graphs. Loading each activity in memory allows the system to realize the links connecting the activities, it also gives the system the ability to immediately process information instead of having to wait for each activity to load from an XML file when it is called. Since each node had changed from the original CxG implementation of the system, the parser had to be reprogrammed. Care was taken in reprogramming the parser so that it would be backwards compatible.

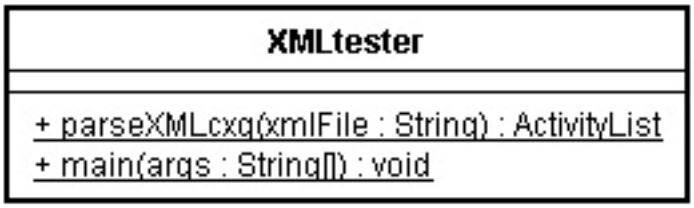

Figure 20: XML Parser Class Diagram

The following classes make up the Inference engine portion of the system. The CxG class is where the logic of the system resides. This class is responsible for stepping

through the graph and processing the inputs presented to the system. The ProcedureContext class is instantiated when a contextual element from the graph is moved from the area of contextual knowledge to procedural context. The Path is a list of the ProcedureContext classes. The following Figures (21, 22, 23, and 24) illustrate the cxgInference package and the subsequent class diagrams of the package components.

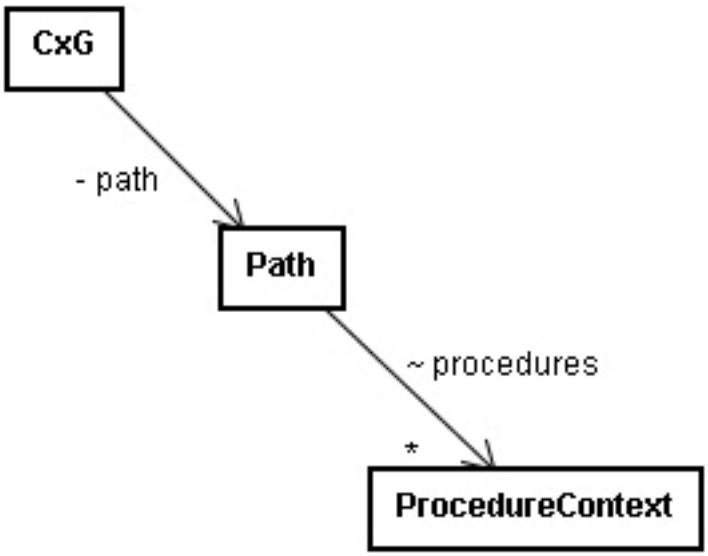

Figure 21: cxgInference Package Diagram

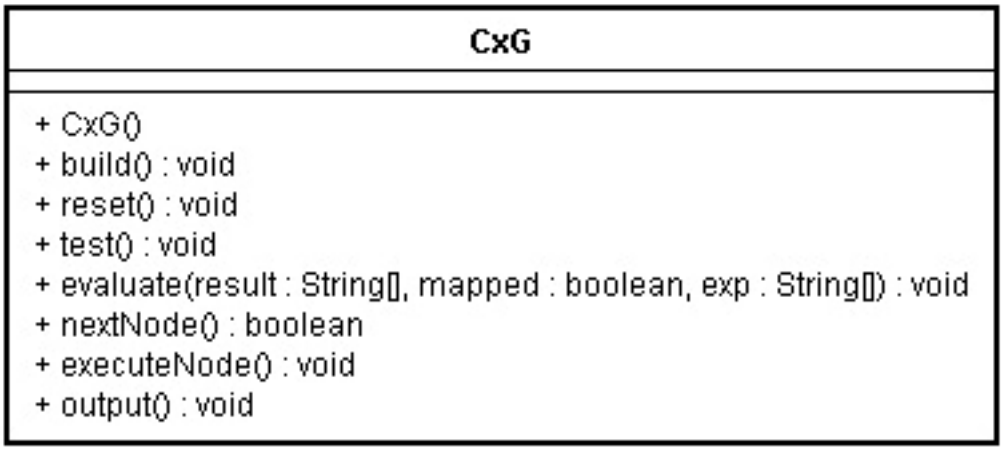

Figure 22: CxG Class Diagram

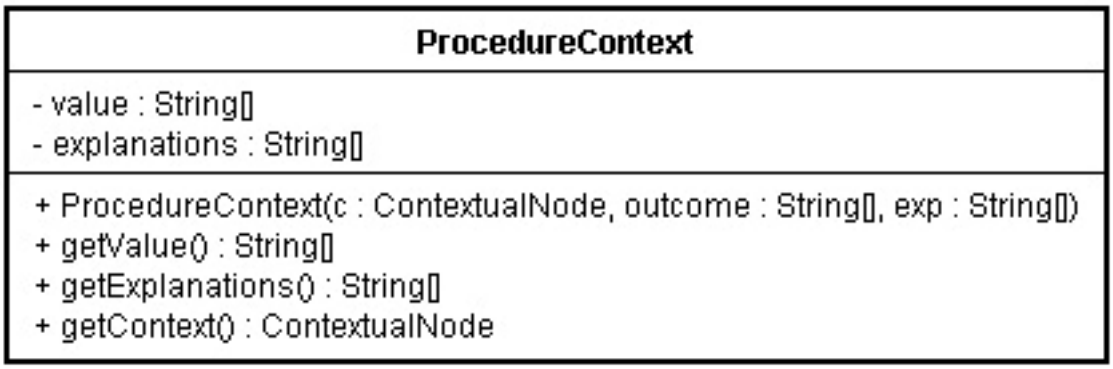

Figure 23: ProcedureContext Class Diagram

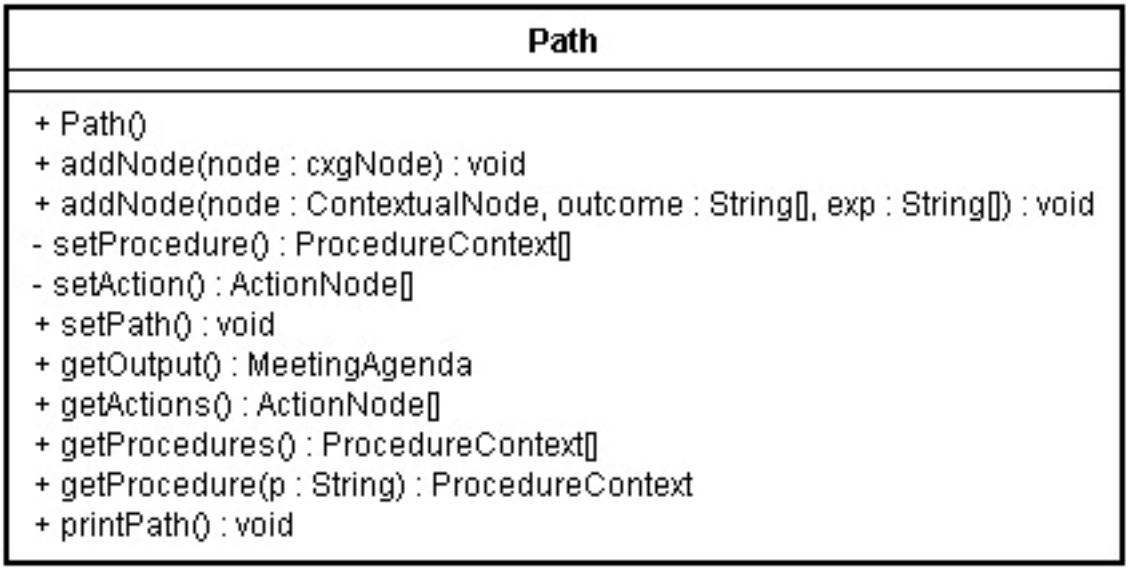

Figure 24: Path Class Diagram

User Interface

The user interface for the AlexDSS prototype was designed to present a series of screens that would present the user with the information contained within the contextual elements. The interface is generic in the respect that it presents the information passed

from the reasoning engine without regard for the knowledge it contains. In developing the interface in this manner, it further provides a separation of the inference portion of the system from the knowledge base.

The user interface is designed to pass information to the user when information needs to be obtained or presented. In the case of the reasoning engine requesting information, upon the user entering the data, the interface sends the input directly back to the engine. When the user interface encounters a contextual or an action node, the pre-explanation contained in each node is presented to the user upon display of the node. As the user progresses through the graph and encounters an action node, the user interface allows the user to display a how-explanation for why that particular action was outputted. After the user completes their session, when progression through the graph is completed, the user interface allows the user to display the cognitive-explanation of the path they have taken through the graph.

In the prototype version of the system, the post-explanations are implemented in the user interface class. The post-explanation was originally designed to be presented only upon user request. Therefore, the functionality of the post-explanation features was implemented within the user interface for convenience to the programmer.

When the how-explanation is called from the GUI, the path is retrieved from the inference engine. A function was programmed into the Path class that allows the retrieval of the active procedural contexts. The explanations are then extracted from the retrieved nodes and presented by the GUI, with the corresponding proceduralized contexts.

For the cognitive-explanation, the entire path is retrieved by the system when the user has completed their entire run through the contextual graph. The path is then parsed

to retrieve the actions that were taken. The how-explanation is then compiled for each of the retrieved actions. The cognitive-explanation is then compiled and formatted by the user interface and outputted onto a terminal screen.

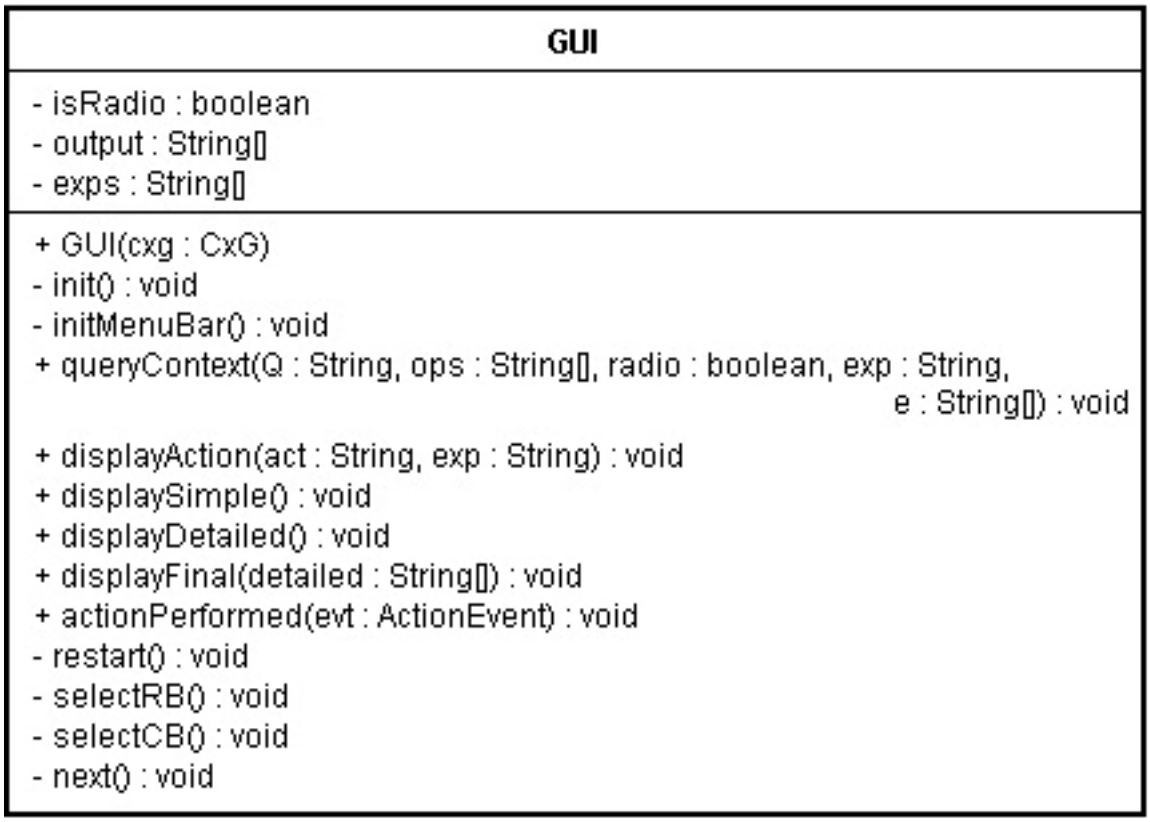

Figure 25: GUI Class Diagram

The following screenshots (Figures 26, 27, 28) from the AlexDSS prototype illustrate the abilities of the user interface discussed in the preceding paragraphs. The screenshots are in sequential order, in the case of a system walkthrough. It will hopefully give an idea of how the system progresses through the knowledge with inputs given from the user. Words highlighted in red and placed below a system query are pre-explanations given by the system.

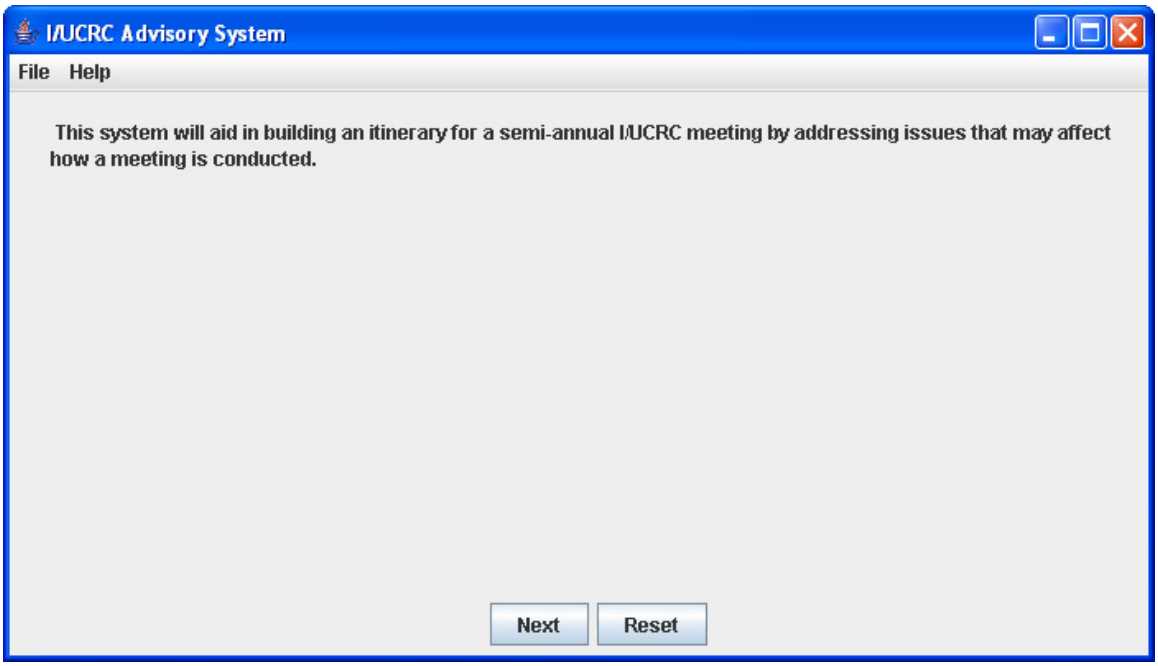

Figure 26: Screenshot of Opening Dialogue

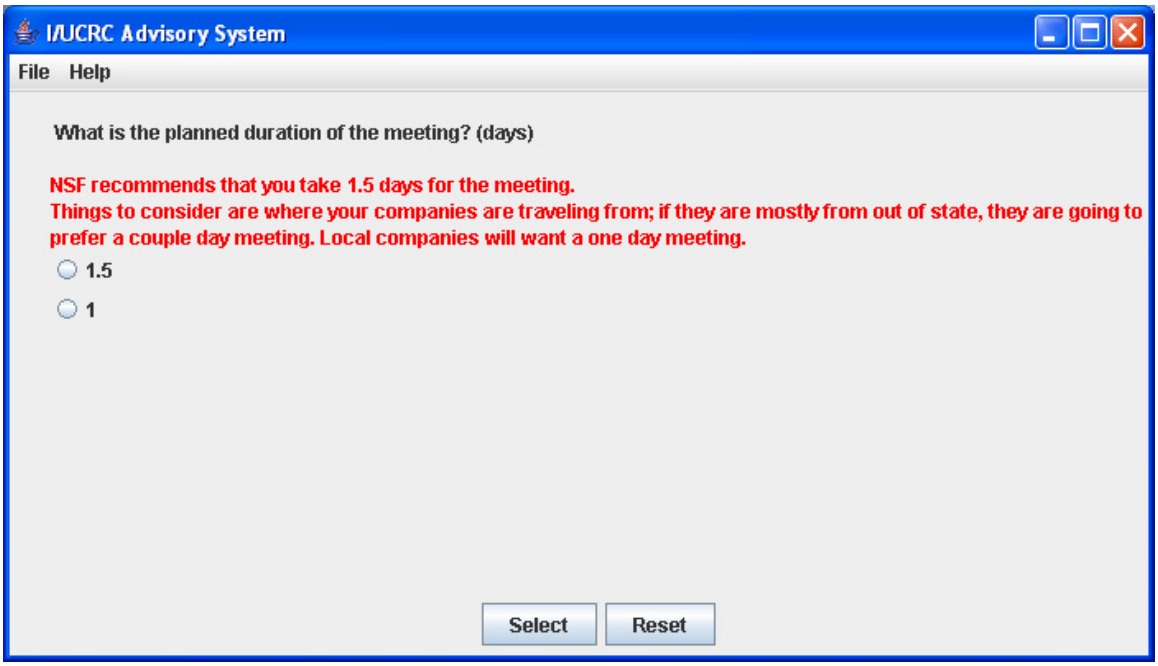

Figure 27: Screenshot of Input Request from User

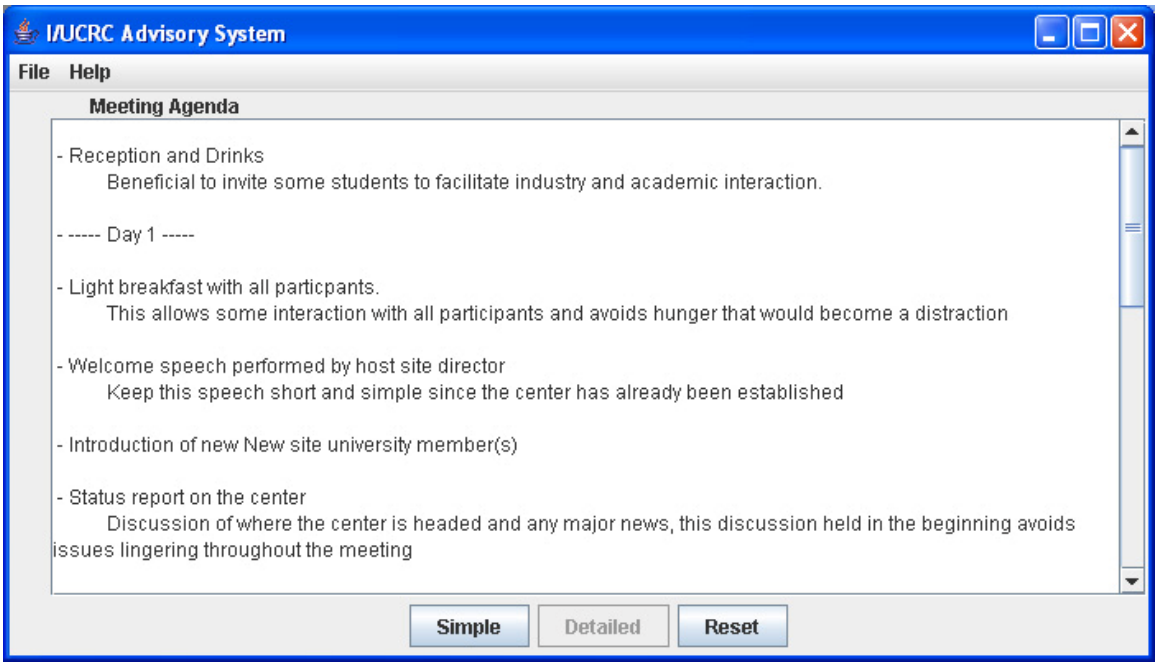

Figure 28: Screenshot of Final Output of the System

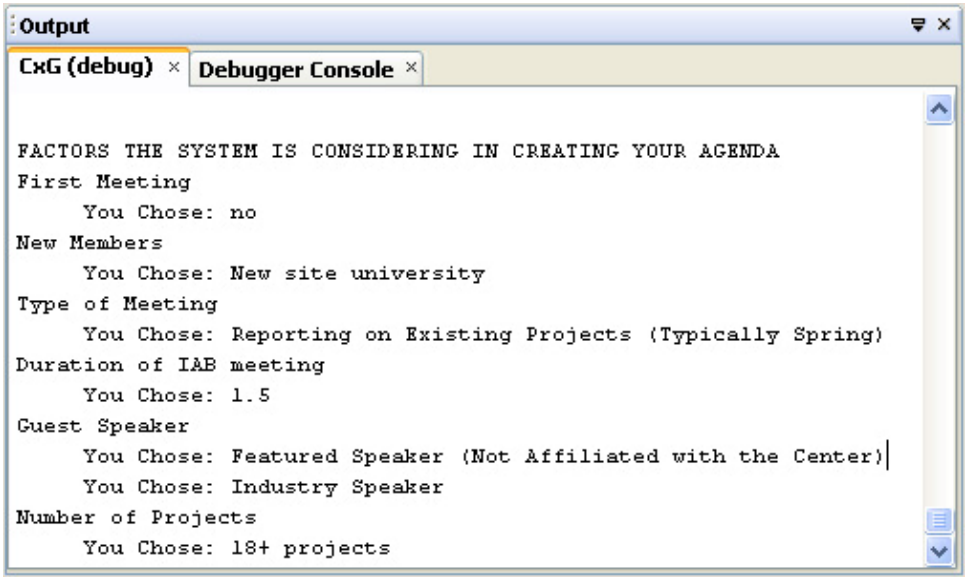

Figure 29: Screenshot of Cognitive-Explanation Output

Figure 28 is the final output of the system. In the case of the AlexDSS prototype, the output is in the form of a meeting agenda to be used for an I/UCRC meeting. As illustrated in Figure 27, the Pre-explanation for this specific contextual node is presented to the user in order to help them make a decision on which path to take. In the prototype this is presented in red text in order to separate it from the standard system dialogue referring to the traditional representation of contextual graphs.

The output of the prototype system is divided into two different output windows, the previously illustrated java GUI and a textual output window. This output window changes based upon the user's current station within the contextual graph. The output window is responsible for displaying post-explanations to the user. While the user is stepping through the process, the output window displays the how-explanation for the user's current point in the contextual graph. When the final output is produced by the system, in this case a *Meeting Agenda*, the cognitive-explanation of the systems activities is displayed for the user. This is illustrated in Figure 29. This allows the user to see exactly what contexts went into the systems overall reasoning process when developing the output.

CHAPTER 5: PILOT EVALUATION

The purpose of the pilot evaluation is to produce initial, albeit preliminary, results on the usefulness of the explanation facility developed here. A separate evaluation, outside the scope of this thesis, was conducted on the initial system without the explanation facility to verify its correctness. These results will be published in a future report and are summarized in the following section. The evaluation done on the prototype system, with the explanation facility, is concerned with determining the usefulness of the explanations presented. They are evaluated on their ability to help the user progress through the graphs, and their ability to help the user understand how the system determines the output presented. These preliminary results will be used to help guide a more complete evaluation process left for future research.

Results of Prototype Testing without Explanations

Knowledge acquired directly from the expert resulted in a transcript similar to a conversation about the topic. Analysis of the knowledge was used to partition it into contexts. Representing the knowledge in Contextual Graphs allows one to meaningfully organize the knowledge into distinct contexts. New knowledge is easily through using the learning ability of CxG discussed in Chapter 3. An example is the identification of the contextual element C2 in Figure 30. The action that results from top branch of this context had originally been thought as only occurring under one context. Upon presenting the contextual graph to the Subject Matter Expert (SME), the SME identified an additional contextual element to be introduced for the bottom branch of contextual element C0.

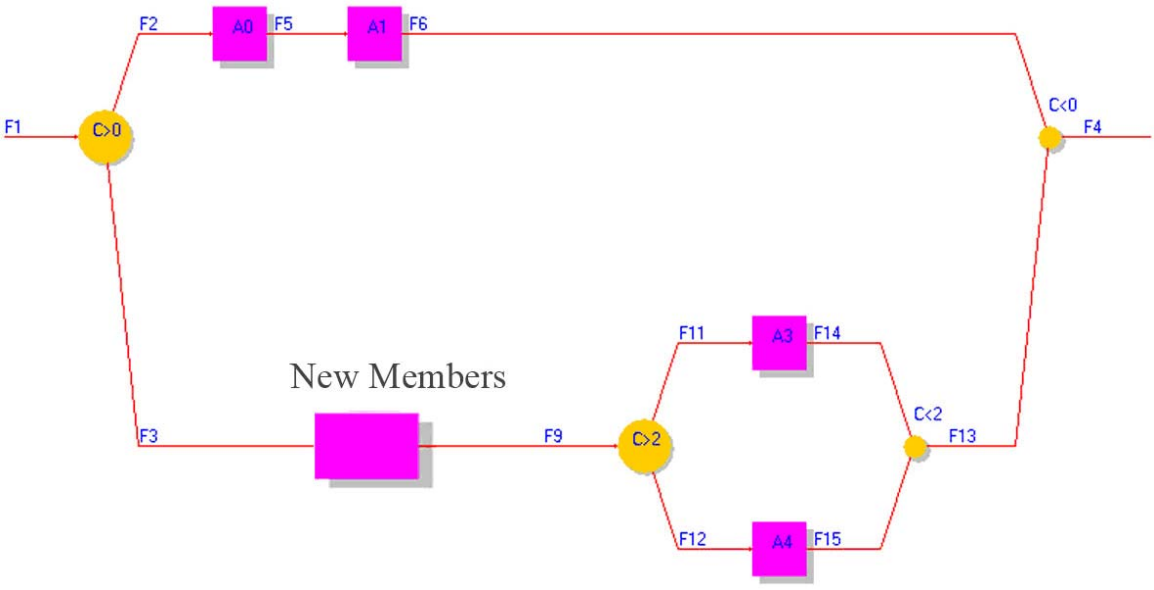

Figure 30: Meeting Agenda CxG

In each situation where a contextual node is presented, the SME would be presented with the possible choices. The response given by the expert was then crosschecked with the contextual graph to assure accuracy of the knowledge representation. When a situation was presented and the expert did not agree with the choices available, a new contextual element was incorporated into the graph. This illustrated the learning capabilities of Contextual Graphs discussed in Chapter 3. An example of this is presented in the following paragraph.

Contextual Graphs enforce the separation of knowledge from the reasoning mechanism. This is especially useful when adding or altering the graphs. Once the reasoning engine was developed, the developer's only concern is to build the graphs. AlexDSS uses XML to quickly assemble and revise the knowledge. This was the case when the activity *New Member* had originally been excluded from activity *Type of Meeting*.

This example of moving the New Member activity into the Type of Meeting, illustrates the ability to alter the knowledge without altering the inference engine of the system. Only the XML file was altered. When the system was re-initialized, the contextual graphs were constructed by the XML parser to incorporate the new knowledge. The inference engine then operated using the newly acquired knowledge. Through the ability to alter the AlexDSS knowledge by updating an XML file, we have successfully separated the knowledge from the system's reasoning.

This example of moving the *New Member* activity into the type of meeting activity displays CxG's effective built-in property of reducing redundancy. The original knowledge acquired for the prototype had this activity independent of the type of meeting activity. Actions of the two activities closely resembled each other, which then identified that the *New Member* activity would have been redundant if the top branch had been taken in C0. Likewise, the explicit declaration of contexts in CxG through branches ensures that contextual elements and activities would only execute in the right situation. In order to proceed down a specific path emanating from a contextual node, the context for that path must be instantiated.

Once the main activity had been accomplished, the system compiles the actions realized along the path taken to form an output. This is important step because the practice includes many static events such as scheduling a meal activity (lunch), but inclusion of it is required in the output. The output of the prototype displays the practice as an itinerary in two forms. The simplified version lists the proceedings throughout the meeting. The descriptive version also adds a description to all proceedings including

static events. This provides insight into the structure of a meeting that is normally provided by the expert.

Verification of the results against the output produced by the Subject Matter Expert was achieved through presenting the final output of the system (the compiled Meeting Agenda) to the SME for approval. The meeting agenda and the contexts that were proceduralized during the creation of the agenda were presented to the SME for evaluation. The contextual graph that the system used was modified based on the SME responses until the output of the system agreed with the output desired by the Subject Matter Expert.

Prototype Pilot Evaluation with Explanations

The following sections describe the pilot evaluation of the explanation facility of the system. For the purpose of clarity, the criteria used for the evaluation is described. The individuals used to evaluate the system are described and the results of their evaluations presented. Following that formal evaluation of the system, personal observations related to the performance of the AlexDSS prototype are presented. Finally, to ensure for an effective and efficient explanation, the system's explanations are evaluated against the five aspects of a good explanation outlined in Chapter 1.

Pilot Evaluation Criteria

The author was responsible for the testing of the system to ensure that the inclusion of the explanation facility did not interfere with the initial prototype. This needed to be established before the formal testing of the system by the evaluators could take place. Because, the contextual graph used was not extensive, it was possible to run through all

possible paths within the graph to determine whether the inclusion of explanations altered the outcome of the system. All possible paths were processed using the original prototype and crosschecked with the corresponding path from the prototype with the explanation facility. It was observed that there were no alterations to the outcome; this was to be expected because the explanations were designed to not interfere with the decision making process. All contextual graph components used by the system were implemented in the contextual graph utilized by the AlexDSS prototype.

The explanation facility was designed to help the human user and not the computer system. The functions incorporated into the classes, for the explanations, are not called upon by the inference engine to process through the contextual graph. Therefore, no matter the configuration of the contextual graph, the explanation facility is unable to influence the way the inference engine processes the knowledge.

The formal criteria of the pilot evaluation involves whether the explanation given is adequate to give the user sufficient understanding of the logic used to reach the decision. This pilot evaluation is also concerned with establishing whether the pre-explanation or post-explanation is more adequate in increasing the user's understanding of the system.

After the evaluator used the afore-mentioned prototype to develop a meeting agenda for an I/UCRC meeting, they were required to answer a series of questions that examined the effectiveness of the explanations produced by the system. This questionnaire also requested their personal observations for improvements and/or modifications that should be introduced to the final system. The questionnaire rated the different responses on a 1 to 5 scale. The following are the meaning of the ratings as

explained to the evaluators: 5 - very useful, 4 - somewhat useful, 3 - no contribution, 2 - not useful, 1 - misleading. The questionnaire given to the evaluators is presented on the following page.

Questionnaire for the AlexDSS Prototype Explanations

Please check only one response for each question.
The scale goes from: 1 = Negative Response to 5 = Positive Response

How useful was the PRE explanation in helping navigate the decision making process?

__ __ __ __ __
1 2 3 4 5

How useful was the PRE explanation in increasing your understanding of how the system reached its decision?

__ __ __ __ __
1 2 3 4 5

How useful was the POST explanation in helping navigate the decision making process?

__ __ __ __ __
1 2 3 4 5

How useful was the POST explanation in increasing your understanding of how the system reached its decision?

__ __ __ __ __
1 2 3 4 5

Which of the explanations, PRE or POST, was more useful in increasing your understanding of how the system reached its decision?

_____ _____
PRE POST

Figure 31: Questionnaire Used in Pilot Evaluation

Evaluators

Three evaluators were used to evaluate the explanation facility of the AlexDSS prototype. Evaluator #1 is a Program Manager for the Industry/University Cooperative Research Center at the National Science Foundation. He has experience in a broad range of engineering disciplines, in part due to his ten years as Vice President of research at a major US University. He has worked as an I/UCRC Program Manager for the last two years and as such is familiar with the knowledge base contained within the system. Evaluator #2 is an Engineering Program Manager dealing with undergraduate education as well as special projects dealing with education in nano-technology. She is familiar with the I/UCRC program and has had over twenty years of experience in managing scientific programs at the NSF. Evaluator #3 is a graduate student in Computer Engineering at the University of Central Florida. He is also a co-developer of the initial AlexDSS prototype and has extensive knowledge of the Contextual Graph representational paradigm.

Results

The proceeding paragraphs discuss the pilot evaluation and preliminary observations gathered on the prototype of the system. The evaluation of the usefulness of the pre-explanation and post-explanation, as well as determining which of the two is more useful, is of primary importance. Following that are observations on the interaction with the system as well as improvements suggested by the evaluators.

The evaluators were first asked to determine the usefulness of the pre-explanation with the following two factors in mind: “How useful was the pre-explanation in helping

to navigate the decision making process?" and, "How useful was the pre-explanation in increasing your understanding of how the system reached its decision?" The pre-explanation received the highest marks on the first question, scoring a 5 from all the evaluators. All testers considered the explanations to be helpful in understanding what the system was asking for, as well as why the system was asking the current question. On the second question concerning whether the pre-explanation increased the evaluators understanding of how the system reached a decision, the pre-explanation received a rating of [5, 4, 4]. As indicated by the results, the pre-explanation is clearly shown to help the user understand how the system processes the knowledge in order to form an output.

Table 2: Preliminary Results for Pre-Explanations

	Evaluator #1	**Evaluator #2**	**Evaluator #3**
How useful was the PRE explanation in helping navigate the decision making process?	**5**	**5**	**5**
How useful was the PRE explanation in increasing your understanding of how the system reached its decision?	**5**	**4**	**4**

The post-explanations were evaluated in the same manner as the pre-explanations; the same two questions were asked and the same rating scale was used to determine their usefulness. In the category of usefulness towards navigating the decision making process, the post-explanation received the rating of [5, 4, 3]. For the category of increasing the evaluator's understanding of how the system reached its output, the post-

explanation received the ratings of [5, 4, 3]. These results indicate that the post-explanation, like the preliminary pre-explanation, was helpful in increasing the users understanding of the system.

Table 3: Preliminary Results for Post-Explanations

	Evaluator #1	**Evaluator #2**	**Evaluator #3**
How useful was the POST explanation in helping navigate the decision making process?	**5**	**4**	**3**
How useful was the POST explanation in increasing your understanding of how the system reached its decision?	**5**	**4**	**3**

The questionnaire also asked the evaluators to determine which of the explanations was more useful in increasing their understanding of how the system reached a decision. Unanimously, all the evaluators chose the pre-explanation as being more useful that the post-explanation.

Personal Observations

The system produces procedural explanations, how and why, based on the path taken through the system. An explanation is included in the graph for each path proceeding from a contextual node. When a contextual node is reached and a path is chosen, the cognitive-explanation produces a chain including the 1) actions and contextual nodes within the path, 2) the corresponding purpose-explanation and why-explanation for each node, 3) the decisions made at each contextual node, and 4) the given explanation for that specific path taken from the contextual node.

It was observed that the reason for the contextual node is sufficiently explained through the why-explanation given to the user to help make their decision. The explanation for each path from the contextual node repeatedly took the form, “Because you selected path 3.2,” (3.2 being an example path). This became repetitive in the way that the system would automatically present the fact that the path chosen was 3.2. An explanation saying, “because you chose 3.2,” was repetitive and proceeded to create a confusing scenario where the system creates repetitive information. Modifications to the explanation facility to prevent this are discussed in the next chapter.

Evaluation Against Aspects

To determine the overall quality of the explanations, the explanations produced by the system are evaluated against the aspects of a good explanation outlined previously (Swartout & Moore 1993). Those five aspects are: Fidelity - the explanation is an accurate representation of what the system does; Understandability - the content of the explanation is understandable to the user; Sufficiency - the explanation provides enough information to satisfy the user; Low Construction Overhead - the explanation poses only a light load on the construction of the system; Efficiency - the addition of the explanation facility does not hinder the performance of the system.

Since the explanation produced is based along the path taken through the contextual graph, the explanation gives an accurate representation of how the system reached its output. This assures the fidelity of the explanations because the logic used to create the explanation is an accurate representation of what the system does. Because fidelity is concerned with presenting an explanation based on the reasoning of the system,

it is evaluated against the cognitive explanation produced. The cognitive explanation presents the chain of events that leads to the output produced. The explanation is a combination of the why explanations of each contextual node, as well as the choice taken by the user. Each action taken is put into context by presenting the procedural context for where the action is executed. Because the explanation given includes all the factors that went into the system's decision, and the sequence in which the knowledge was inputted and processed, the fidelity of the explanation is preserved.

Understandability of the explanation is determined, given that the fidelity aspect of the explanation is acceptable, to a large extent by the grammar and language used in producing the knowledge to be incorporated in the contextual graph. An effort was made to use plain and simple statements when developing the graph used in the prototype system. These statements would be easily understood by someone not familiar with the intimate details of the knowledge referenced by the content in the graph. All evaluators expressed, with maximum confidence, that they had no difficulty in understanding the statements produced by the system. As stated previously, Understandability is concerned mostly with the way in which the knowledge entered into the system is phrased and constructed.

Explanations produced by the system were modeled after actual responses given by the subject matter expert that the system was designed to represent. To this extent, the explanations given by the system are Sufficient enough to account for being a good explanation. Modifications to the prototype to improve the sufficiency are discussed the proceeding chapter under the section of future work.

According to the observations of the author, the inclusion of explanations into the knowledge within the system is an easy and straightforward process. Low construction overhead is observed by the fact that the “why” or “purpose” of an action or decision node is known by the knowledge engineer when construction the graphs. The explanation knowledge is included by inserting an extra tag within the corresponding node in the XML file used to build the system. When the knowledge is gathered for a certain situation the knowledge engineer is assumed to know the logic that went into the decision making process. Knowledge is not gathered blindly nor in a vacuum, the engineer is aware of the situation in which the activities take place. Therefore, including this into the graph is not an arduous process, nor should it be one that is time consuming since the knowledge for said events is at hand when constructing the system. The cognitive and procedural explanations are generated by the inference engine of the system, therefore the knowledge engineer need not be concerned with the development of these types of explanations.

Efficiency of the system is not affected by the inclusion of the explanation facility. From the perspective of the inference engine, the explanations of the system are not involved in the processing of the user through a contextual graph. The explanations for many of the nodes are presented with no computation. The explanations requiring proceduralized context are processed in an efficient manner. The path that the user has taken through the system is recorded and therefore, the search space for the development of an explanation is limited only to the knowledge accessed by the user and not the entire knowledge base. If the run-time efficiency is degraded, it is only done so in such a minimal manner that is not observable to a human observer.

Conclusions

A pilot evaluation of the system was done before the explanation facility was added to the prototype to show that the system was able to correctly represent and process through a contextual graph. The output of the system was cross referenced against the Subject Matter Expert and modified until the output produced was approved by the SME as being correct.

The pilot results of the evaluation of the explanation facility show that the pre-explanation is preferred over the pot-explanation in helping the user understand the logic the system uses to reach a decision. It is noted that even though the post-explanation scored lower than the pre-explanation, in the evaluation, the post-explanation was still found to be helpful in increasing the user's understanding of the system. The explanations produced were also evaluated against the 5-aspects of a good explanation outlined in Chapter 1. It was shown that the explanations satisfied the criteria set forth by having fidelity, understandability, sufficiency, low construction overhead, and efficiency.

CHAPTER 6: SUMMARY, CONCLUSIONS & FUTURE WORK

The following sections summarize the research conducted and the work done within this thesis. Also, conclusions are drawn based upon the results of the system's pilot evaluation. Future work, concerning future extensive testing of the system as well as modifications to the explanation facility, are discussed in the final section.

Summary

It has been established that knowledge-based systems must do more than simply answer a user's question; they must be able to explain their answer in a manner that is acceptable and understandable to the user. It has been shown that with the increase in understanding of a knowledge-based system by the user, there is more acceptance of the system and henceforth, the system is utilized more frequently. A way in which to achieve increased understanding from the user's perspective is to have the KBS explain its reasoning and output to the user.

The idea of introducing explanations into intelligent system is not new. There has been a considerable amount of work employed to attempt to produce a valid explanation to the end user that is both understandable and reflective of how the actual system behaves. Early work to facilitate explanations in rule-based systems, such as NEOMYCIN, were successful in the fact that their systems were able to output explanations. However, these explanations required an additional intelligent system to operate in conjunction with the original system to produce a viable explanation. The use of meta-rules was employed to help decipher the rules that were responsible for producing the output. Thus, separating the explanation from the logic the system used to

compute the output. Case-based systems also produced explanations that were considered viable and sufficient in a limited form. The explanations presented were often references to other cases, or in some instances cases themselves. More complex explanations were able to be produced by employing a hybrid system that used rule-based or model-based inferencing. As in previous attempts to explain intelligent systems, this type of explanation also separated the logic used to compute the system's output from the logic used to explain how the system produced the output.

A Contextual Graph is a knowledge representation paradigm that allows for the separation of knowledge into specific contexts. The CxG paradigm has multiple benefits for knowledge representation. The search through the knowledge space is limited to only those contexts that apply; this prevents the system from asking of irrelevant questions that are not pertinent to the current situation. The unique construction of contextual graphs allows for an explanation to be produced that is both efficient and sufficient.

There are two temporal formats in which the explanations are presented: a pre-explanation and a post-explanation. The pre-explanation is presented to the user before information is required to be inputted. This type of explanation is intended to help the user navigate the decision making process and to justify the query presented by the system. The post-explanation is designed to be presented to the user upon the execution of an action by the system. This allows for the system to explain exactly what context(s) went into the decision making process that led to the execution of said activity.

The pre-explanations are divided into two categories: a purpose explanation and a why-explanation. The purpose explanation describes the purpose of an action and is not restricted to the context in which it appears. This was done in order to preserve the ability

of contextual graphs to reuse actions within a graph. The why-explanation is specific to the contextual node it is attached to. The why-explanation is designed to explain why the question is being asked and assist the user in making a decision on which path to proceed from the contextual node.

Like the pre-explanations, post-explanations are also divided into two separate categories: how-explanations and cognitive-explanations. The how-explanation is considered to be a form of a why-explanation. The how-explanation is procedural in nature and describes the context in which an action took place. The cognitive-explanation is similar to a how-explanation in the way that it also procedural in nature and describes why an action took place. However, the cognitive-explanation is a description of the entire path taken by the user through the system and describes why all the actions executed took place. The cognitive-explanation is presented to the user at the end of a session when the final answer has been reached.

Through preliminary evaluation of the system, it was shown that the pre-explanation was considered to be more useful than the post-explanation in helping the user navigate the decision making process. The pre-explanation was also shown to be more useful than the post-explanation in increasing the user's understanding of how the system reached its decision. When given the choice of which explanation was considered more useful in increasing their understanding of the system, the evaluators unanimously chose the pre-explanation over the post-explanation. It is noted, however; that according to the ratings, the post-explanation was still considered to be useful in increasing their understanding of the system. Therefore, we conclude that it is advisable to keep both

types of temporal explanations in the system. Modifications to improve the explanation facility with accordance to the temporal formats are discussed in the following section.

Conclusions

The following is the list of contributions discussed in Chapter 2 and realized by this thesis:

1. An explanation facility was added to the Contextual Graph Paradigm that was shown to be effective through the pilot evaluation done on the prototype system.
2. The pre-explanation was shown to be more effective than the post-explanation, in the pilot evaluation, in helping the users understand the system's decision making abilities.
3. A prototype system called the AlessDSS was implemented, using Contextual Graphs ,that was able to explain its decision making abilities.
4. Test data was produced from the pilot evaluation that shows the results of the explanation testing.

As shown in the pilot evaluation, the incorporation of the explanation facility into the contextual graph paradigm increased the user's understanding of how the system functions. This validates the first condition set forth in the hypothesis. Also shown in pilot evaluation, the pre-explanation was considered the more valued explanation for helping the user navigate the decision making process. This satisfies the second condition set forth in the hypothesis presented. To this extent, the how-explanation discussed in Chapter 2, should be migrated over from the post-explanation to the pre-explanation.

Currently, the how-explanation executes when an Action is executed by the system, showing the user exactly what contexts went into the system's decision to execute said action. The author believes that the how-explanation should be presented at the time of a query being presented to the user by the system. Currently, the why-explanation is presented to the user along with the query from the system. This is done in order to help the user make a decision on which path to take from the current contextual node, this has shown to be a good way in which to help the user navigate the decision making process. By presenting the how-explanation at this point, it is proposed that this will give the user a better understanding of why the system is asking the question in the first place. It should accomplish this by displaying to the user exactly what context(s) placed the user at their current point in the contextual graph. The context(s) that led to an action being executed will still be shown through the cognitive-explanation.

The post-explanation was shown to be the equal of the pre-explanation, within a 0.33 scored margin, in helping the user understand why the system reached its decisions. The post-explanation scored and average of 4 in the pilot evaluations where the pre-explanation scored an average of 4.33. This author believes that the strength of the post-explanation is in the area of helping the user understand how the system reached a decision. By removing the how-explanation from the post-explanation category to the pre-explanation category, this leaves only cognitive-explanations to be presented after the user has made their decision(s). The author believes that a cognitive-explanation is an excellent way to present a record of the system's decision making process. To this effect, a facility will be implemented in the final AlexDSS to give the ability to the user to print their final output along with the cognitive-explanation for the output. This should provide

the user with a record of the system's decision as well as a quick reference they may use to refer to a previously asked query. By presenting the post-explanation in this manner, it is believed that the users of the system will appreciate the post explanation more in helping them understand the logic used by the system to reach its decision(s).

It is believed that the practicality of implementing these explanations into a system that uses contextual graphs to represent its domain knowledge is fully realizable. If the knowledge to be represented is able to be expressed in the form of contextual graphs; then do to the evaluation against the 5-aspects of a good explanation presented in Chapter 5, the addition of explanations into the system should be possible and practical for any real world system using Contextual Graphs. Likewise, do to the pilot evaluation of the pre-explanation and post-explanation, the explanations produced should be applicable for any domain knowledge that can be represented in Contextual Graph form.

Future Work

A good user interface is important for any system that needs to interact with a human user. Knowledge Based Systems are designed to replicate advanced knowledge in order to present that information in a form that is acceptable to a user that is not especially proficient in the domain area that the system covers. Therefore, work is continuing on a more advanced Graphical User Interface that is engaging as well as being efficient in the way it presents the information to the user.

As noted previously, the prototype divides the explanation feature into two separate interfaces: pre-explanations are presented on the same screen as the context queries, and post-explanations are presented on a text based terminal output window. The

post-explanations will be incorporated into the graphical interface, this will allow for more advanced formatting in order to make the explanation graphically appealing to the user. As indicated previously, the how-explanation will be moved over into the pre-explanation category. This change will be reflected in the User Interface.

During the writing of this thesis, Graphical Interface work has been continuing by the ISL team at the University of Central Florida. For the final version of the AlexDSS, a server client model has been implemented. The inference engine is run on a java server with the client side interaction being implanted using a Flash interface. Using flash allows for much greater flexibility in producing a more engaging graphical user environment.

A future evaluation of the system involving a larger test group of evaluators as well as a more comprehensive questionnaire is planned to take place within the next year. Knowledge, in the form of contextual graphs, is being added into the AlexDSS system in order to make it a more robust Decision Support System able to answer any question within the Subject Matter Expert's domain. This knowledge, combined with the server-client model, and the improved graphical user interface; allows the users of the system to conduct a more exhaustive testing over multiple contextual graphs. In this way, the effectiveness of the pre-explanations and post-explanations can be determined based upon multiple topics within the knowledge domain.

REFERENCES

P. Brézillon, "Representation of Procedures and Practices in Contextual Graphs," *Knowledge Engineering Review*, 2004.

P. Brézillon, "Context Dynamic and Explanation in Contextual Graphs," *CONTEXT 2003*, LNAI 2680, 2003a, 94-106

P. Brézillon, "Context-based Modeling of Operators' Practices by Contextual Graphs," *Human Centered Process: 14th Mini Euro Conference, Luxembourg,* 2003b.

P. Brézillon, "Modeling Decision Making with Context-Based Reasoning and Contextual Graphs. Application in Incident Management on a Subway Line," *Human Centered Processes 99, Brest, France*, 1999.

P. Brézillon, "Contextualized Explanations," *International Conference on Expert Systems for Development*, Bangkok, Thailand, March 1994

G. Carenini, J.D. Moore, "Generating Explanation in Context," *Proc. of the International Workshop on Intelligent User Interface*, Orlando, 1993.

W. Clancy and R. Letsinger, "NEOMYCIN: Reconfiguring a Rule-Based Expert System For Application to Teaching," *Stanford University Report, STAN-B-82-908,* May 1982.

P. Cunningham, D. Doyle, and J. Loughrey, "An Evaluation of Usefulness of Case-Based Explanation," Technical Report, TCD-CS-2003-41, Trinity College, Dublin 2003.

D. Doyle, A. Tsymbal, and P. Cunningham, "A Review of Explanation and Explanation in Case-Based Reasoning," Technical Report, TCD-CS-2003-41, Trinity College, Dublin 2003.

A. J. Gonzalez, R. Ahlers, "Context-based Representation of Intelligent Behavior in Training Simulations," *Transactions of the Society for Computer Simulation International*, Vol. 15, No. 4, pp. 153-166, March 1999.

A. J. Gonzalez, D. D. Dankel, "The Engineering of Knowledge Based Systems: Theory and Practice," *Prentice Hall, Englewood Cliffs, NJ*, 1993.

J. Herlocker, J. Konstan, and J. Riedl, "Explaining Collaborative Filtering Recommendations," *Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work,* 2000.

H. C. Lane, M. Core, M. van Lent, S. Solomon, & D. Gomboc, "Explainable artificial intelligence for training and tutoring," *To appear in Proceedings of the 12th*

International Conference on Artificial Intelligence (AIED05), Amsterdam, The Netherlands, 2005.

L. Karsenty, P. Brézillon, "Cooperative problem solving and explanation," *International Journal of Expert Systems With Applications 4* pp 445-462, 1995.

D. Leake, "Case Based Reasoning: Experiences, Lessons, and Future Directions," *Menlo Park: AAAI Press/MIT Press*, 1996, Chapter 1.

M. Macedonia. "Ender's Game Redux," *Computer*, Vol. 38, no. 2, pp. 95-97, February 2005

J. Pomerol, P. Brézillon, "About some relationships between Knowledge and Context," *CONTEXT-01,* Lecture Notes in Computer Science. Springer Verlag. Dublin 2003.

D. Richards, "Knowledge-Based System Explanation: The Ripple-Down Rules Alternative," *Knowledge and Information Systems*, Volume 5, Issue 1, March 2003, 2-25.

T. Roth-Berghofer, "Explanations and Case Based Reasoning: Foundational Issues" *Advances in Case-Based Reasoning, ECCBR,* 2004.

R. Schank, "Explanation Patterns: Understanding Mechanically and Creatively," *Lawrence Erlbaum Associates, Hillsdale, NJ,* 1986.

E.H. Shortliffe, "Computer Based Medical Consultations: MYCIN" *Elsevier, New York,* 1976.

P. Spieker, "Natürlichsprachliche Erklärungen in technischen Expertensystemen," *Ph.D. Dissertation, University of Kaiserslautern,* 1991.

J.F. Sowa, "Representing and reasoning about contexts," *Proc. of the AAAI'92 Workshop on Propositional Knowledge Representation,* Stanford, CA pp 133-142, 1992.

W. Swartout, J. Moore, "Explanation in Second Generation Expert Systems," *Second Generation Expert Systems, Berlin, Springer-Verlag,* 1993, 543-585.

Relevant References not Cited

R. Barzilay, D. McCullough, O. Rambow, J. DeCristofaro, T. Korelsky, B. Lavoie, "A New Approach to Expert System Explanations," *In Proceedings of the Ninth International Workshop on Natural Language Generation*, pp. 78-87, 1998.

P. Brézillon P “Architectural and contextual factors in explanation construction,” *In Proc. of the IJCAI Workshop on Improving the Use of Knowledge-Based Systems with Explanations,* Vienna, Austria. Research Report 92/21, LAFORIA, University of Paris 6, France. 65-74, 1992.

W.J. Clancey, “The epistemology of a rule-based expert system: A framework for Explanation,” *Artificial Intelligence Journal 20(3),* pp 197-204, 1983.

L. Karsenty, P. Falzon, “Spontaneous explanations in cooperative dialogues,” *Proceedings of the ECAI'92 Workshop on Improving the Use of KBS with Explanation, Technical Report 92/21*, LAFORIA, University Paris 6, France, June, pp 115-124, 1992.

D.B. Leake, “Evaluating explanations,” *Lawrence Erlbaum Associates Inc,* 1992

B. Mark, “Explanation and interactive knowledge acquisition,” *Proceedings of AAAI'88,* Workshop on Explanation, 1988.

W. R. Swartout, C.L. Paris, J. D. Moore, “Design For Explainable Expert Systems,” *IEEE Expert*, Volume 6, Number 3, pages 58-64, 1994.

J. T. Yao, “ Knowledge Based Descriptive Neural Networks,” *Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing,* Chongqing, China, Lecture Notes in Computer Science 2639, pp430-436, 2003.