Implementation of an Ontology for the Evaluation of IED-Disposal Training

by

Michael William Lepard

A thesis submitted in conformity with the requirements for the degree of Master of Engineering

Mechanical & Industrial Engineering

University of Toronto

© Her Majesty the Queen in Right of Canada, as represented by the Minister of Defence, 2011
Implementation of an Ontology for the Evaluation of IED-Disposal Training

Michael William Lepard

Master of Engineering

Mechanical & Industrial Engineering
University of Toronto

2011

Abstract

This project proposes an ontology that can be used to evaluate the gathering of information performed by an IED-Disposal student during training. The ontology uses the activities of asking questions, obtaining and classifying SA elements to propose axioms that control the generation of evaluation comments. Evaluation of the axioms was conducted in first order logic using Prover9 which proved problematic in the development of a real-time decision support system for use in a training simulator. The axioms were migrated to Prolog and implemented in an inference engine as part of the prototype DSS and user interface. Scenarios were developed for the user interface that would both verify the behaviour of the evaluation ontology and evaluate the performance of the DSS under real-time loads.
Acknowledgments

This work was performed for Defence Research and Development Canada (DRDC) through CAE Professional Services. The author would like to acknowledge the support of Dr. Ming Hou at DRDC Toronto and Dr. Michael Gruninger at the University of Toronto.
Table of Contents

Acknowledgments........................................................................................................................................ iii
Table of Contents....................................................................................................................................... iv
List of Tables ............................................................................................................................................... vii
List of Figures ............................................................................................................................................... viii
List of Appendices ....................................................................................................................................... ix
List of Acronyms ........................................................................................................................................ x
Chapter 1 Ontology Description .................................................................................................................. 1
  1 Introduction ........................................................................................................................................ 1
    1.1 Report Structure ................................................................................................................................. 2
  2 Background .......................................................................................................................................... 3
    2.1 Verbal SA Elements ............................................................................................................................. 3
    2.2 Visual SA Elements .............................................................................................................................. 5
  3 Ontology Requirements .......................................................................................................................... 5
  4 PSL – Foundational Ontology ................................................................................................................ 7
  5 Taxonomy ............................................................................................................................................... 8
  6 Logical Axioms ....................................................................................................................................... 10
    6.1 Questioning Evaluation Axioms ........................................................................................................... 10
      6.1.1 Questioning Technique Comment ................................................................................................. 11
      6.1.2 Missed Follow-up Question Comment ........................................................................................... 11
      6.1.3 Questioning Efficiency Comment ................................................................................................ 12
    6.2 SA Element Evaluation Axioms ......................................................................................................... 13
      6.2.1 Wrong SA Element Classification Comment ................................................................................. 13
List of Tables

2-1 Component Word Properties ........................................................................................................... 5

A-1 Intended Model Inputs ................................................................................................................. 45

A-2 Intended Model Outputs ............................................................................................................. 46
List of Figures

2-1 Example Dialog Tree ................................................................................................................ 4

3-1 Questioning Technique Competency Questions ....................................................................... 6

3-2 SA Element Competency Questions ........................................................................................ 7

5-1 Evaluation Ontology Taxonomy .............................................................................................. 9

10-1 - Question Widget ................................................................................................................. 30

10-2 Answer Widget ..................................................................................................................... 31

10-3 Evaluation Feedback ............................................................................................................. 31

10-4 Scene Widget "Reception" ................................................................................................... 32

10-5 Scene Widget "Office IED" .................................................................................................. 33
List of Appendices

A  Intended Models .................................................................................................................. 45

B  Prover9 Input ....................................................................................................................... 47
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CF</td>
<td>Canadian Forces</td>
</tr>
<tr>
<td>CFB</td>
<td>Canadian Forces Base</td>
</tr>
<tr>
<td>DND</td>
<td>Department of National Defence</td>
</tr>
<tr>
<td>DRDC</td>
<td>Defence Research &amp; Development Canada</td>
</tr>
<tr>
<td>DSS</td>
<td>Decision Support System</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>IED</td>
<td>Improvised Explosive Device</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Tutoring System</td>
</tr>
<tr>
<td>NPC</td>
<td>Non-Player Character</td>
</tr>
<tr>
<td>PSL</td>
<td>Process Specification Language</td>
</tr>
<tr>
<td>UDP</td>
<td>Unreliable Datagram Protocol</td>
</tr>
<tr>
<td>SA</td>
<td>Situational Awareness</td>
</tr>
<tr>
<td>WPF</td>
<td>Windows Presentation Foundation</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
</tbody>
</table>
Chapter 1
Ontology Description

1 Introduction

Improvised explosive devices (IEDs) are a major challenge for security forces today. IEDs are usually placed in public areas and detonated to cause damage and loss of life. In order to properly dispose of an IED the response team must first determine how the IED is detonated. IEDs have three methods through which they can be detonated by the bomber: victim, timer and command. In a victim based IED the device is detonated through some action of the victim. Examples of these victim detonation methods can be trip-wires or packages rigged to explode upon opening. In a timer based IED the device is detonated at a specific time specified by the bomber. Some examples of these victim detonation methods can be a count-down timer or a mechanical watch that triggers the detonation at a time of day. Finally a command based IED is detonated through a signal to the device by the bomber. An example of a command detonation method is a cell phone device that triggers when it receives a call.

In order to correctly classify an IED the response team must gather information on the device by questioning witnesses and examining the environment. Important details can be gathered which provides information about the scene and intended targets. This information can then provide situational awareness (SA) elements that are used to classify the method of detonation. Good questioning technique and a keen eye can provide correct and useful SA elements while a bad questioning technique or casual glance can bias the response team and lead to an incorrect classification of the detonation method.

The Canadian Forces has an IED Disposal training course designed to teach these skills to the response team. Defence Research and Development Canada (DRDC), an agency of the Department of National Defence, is leading an effort to develop an intelligent tutoring system (ITS) for use in the course [1]. The student will be presented with a scenario that contains several witnesses at a scene where an IED has been discovered. The student must inspect the scene to obtain visual SA elements and question witnesses to obtain verbal SA elements. The student will classify the SA elements in order to determine the detonation method of the device. The ITS component of the system will evaluate the progress of the student and provide
instructional feedback where appropriate. The ontology presented in this work represents a model of the expert knowledge the student should have and will be used in the evaluation. The implementation of the ontology represents the first attempt at developing the decision support system that will be the basis for the ITS evaluation engine.

1.1 Report Structure

This report documents the development and evaluation of the IED evaluation ontology in four chapters: a formal description of the ontology; a real-world implementation using Prolog; an evaluation of the ontology and finally a summary.

The following is a summary of the contents within the formal description of the ontology:

- A background of the domain which introduces the types of questions that will be evaluated and the concept of situation awareness elements,

- The requirements of the ontology through the introduction of competency questions that will be used to drive the focus of the ontology,

- The selection of a foundational ontology to use as the basis,

- The structure and taxonomy that will be used in the ontology,

- The formal logical axioms of the ontology.

The following is a summary of the contents that document the real-world implementation of the ontology:

- The requirements and resulting selection of a Prolog inference engine,

- The conversion of the first order logical axioms of the ontology into Prolog rules.

The following is a summary of the contents that document the evaluation of the ontology:

- A description of the intended models,

- An evaluation of the first order logical axioms within Prover9,
• A description of a software based decision support system, and associated student interface, that uses the Prolog inference engine.

Finally, the report summarizes the evaluation ontology with the following sections:

• A review of the ontology,

• A conclusion summarizing the steps of the project,

• And a discussion on future work on other domain applications of the ontology.

2 Background

To evaluate a student’s progress in the training scenario the ontology will use various events from the user interface [2]. These events revolve around two types of SA elements: verbal and visual. Each type has several events, or activities, associated with it and the activities have objects and properties that are important to generating an evaluation.

Both types SA elements are considered hidden when the scenario starts, the student does not know about them and is trying to reveal them. Verbal SA elements are revealed by asking questions to a witness, Non-Player Character (NPC), while visual SA elements are revealed by investigating the scene and clicking on significant parts of an image. The methods through which the student obtains these SA elements is a key component of the evaluation ontology. [4]

All SA elements provide clues to the type of trigger mechanism used in the IED and their correct classification is specified by the scenario designer. When the student reveals the SA element they are asked to classify it against one or more of the trigger types. The ontology then evaluates the correctness of the proposed classification and provides feedback if necessary.

2.1 Verbal SA Elements

Verbal SA elements are revealed to the student by selecting questions from a predefined list. Each question is also tagged with information that is used in the evaluation of the student. One role of the ontology is to capture the details of this information and how it is used in the evaluation.
The list of questions available for each NPC is organized in a dialog tree with a unique identification number, using a common parent-child relationship. Figure 2-1 shows an example of the dialog tree and the numbering scheme used to identify a specific question. The numbering scheme makes it easy to identify whether two questions share a parent-child relationship.

![Diagram of a dialog tree](image)

### 2-1 Example Dialog Tree

Questions have two pieces of information that are used in the evaluation: the component word property and the answer type. When choosing questions to ask a witness the student should select questions that provide good information while avoiding questions that might bias or confuse the witness. When evaluating the type of question selected it is important to consider what type component words, such as jargon or double negatives, that make up the question. Table 2-1 describes the types of component word properties that are evaluated in the tutor and provides some example responses it should present to the student when they selected that type of question.

<table>
<thead>
<tr>
<th>Component Word Properties</th>
<th>Tutor Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Good Questions</strong></td>
<td></td>
</tr>
<tr>
<td><em>Precision</em></td>
<td></td>
</tr>
<tr>
<td>e.g. where did you go <em>exactly?</em></td>
<td>Good use of precision words, this will help the witness give you a specific answer.</td>
</tr>
<tr>
<td><em>Open Ended</em></td>
<td></td>
</tr>
<tr>
<td>e.g. Can you describe the backpack?</td>
<td>Appropriate use of open-ended question, this helps to establish rapport with the witness.</td>
</tr>
<tr>
<td><em>Visual Aids</em></td>
<td></td>
</tr>
<tr>
<td>e.g. can you sketch me a diagram of the location?</td>
<td>Good use of visual aids to support the answer.</td>
</tr>
<tr>
<td><em>Funnelling</em></td>
<td></td>
</tr>
<tr>
<td>e.g. Can you tell me more about the bags position?</td>
<td>Good use of funnelling – going from broad to specific can help focus in on particular information</td>
</tr>
<tr>
<td><strong>Bad Questions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Component Word Properties</strong></td>
<td><strong>Tutor Response</strong></td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------</td>
</tr>
</tbody>
</table>
| **Vague**  
e.g. [in response to prior open ended Q]  
Tell me more. | Inappropriate use of consecutive open-ended questions – try to focus in on something more specific. |
| **Leading**  
e.g. So the wires were coming out of the ground? | Careful using leading questions, you may bias the witness’s answer |
| **Double Negative**  
e.g. Did you not go home right after? | Avoid double negatives- ask question directly |
| **Jargon**  
e.g. where are the UXO? | Avoid Jargon that is only familiar to a certain audience- use more general, non-technical terms |
| **Compound**  
e.g. what did you do last night, did you go to the store, when did you come home? | You are asking too many questions at once; ask only one question at a time. |

### 2-1 Component Word Properties

The last piece of information associated with a question is the answer type, which indicates whether the SA element or one of its children reveal an SA element. This information is used within the evaluation to ensure the student is being efficient in their questioning strategy and is not missing a question that reveals important information.

### 2.2 Visual SA Elements

Visual SA elements are revealed when the student clicks on the appropriate image within the scene. The ability and speed to detect objects that are out of place are important factors in the evaluation of the students behaviour. To determine how effective the student is at identifying import visual elements in the scene the evaluation ontology will use a boundary area defined around each visual SA element. The amount of time the student’s mouse is within the boundary, without the student clicking to reveal an SA element, is tracked by the evaluation ontology. When it becomes clear that the student is unable to identify the visual SA element the evaluation ontology will provide an appropriate hint to aid the student.

### 3 Ontology Requirements

The requirements of the IED evaluation ontology will be defined through the use of competency questions based on the background information presented in the previous section. Competency questions provide a mechanism for defining what knowledge the ontology should contain and
what information can be extracted from it. This ontology focuses on two area of the IED disposal process: SA elements and questioning technique.

<table>
<thead>
<tr>
<th>Question Asked</th>
<th>Answer Type</th>
<th>Component Word Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question Asked</td>
<td>Answer Type</td>
<td>Component Word Property</td>
</tr>
</tbody>
</table>

3-1 Questioning Technique Competency Questions

Figure 3-1 shows the desired behaviour of the questioning technique component of the evaluation ontology. Component word and answer type information contained within the questions selected by the student are used to generate different evaluation comments. Section 5 presents a formal description of the circumstances under which each evaluation comment is generated.

Figure 3-2 shows the desired behaviour of the SA element component of the evaluation ontology. During the training scenario the student will discover clues containing the SA elements and classify their trigger type. The mouse curser is also tracked and its position relative to visual SA elements is used in the decision to generate comments. Section 5 presents a formal description of the circumstances under which each evaluation comment is generated.
4 PSL – Foundational Ontology

The Process Specification Language (PSL) was chosen as the foundational ontology due to the similarity of concepts between the two ontologies. At its core the PSL ontology uses activities, unique occurrences of those activities and objects participating in those occurrences to formally specify the details around complex processes. The document *Using the PSL* [3] provides an excellent introduction to the PSL ontology and how it can be used to specify processes. Although the PSL is aimed at manufacturing processes the concepts of asking questions, interacting with SA elements and generating comments align well with the concept of activities occurring within the PSL. The specific comments, questions and SA elements that participate in these activities are mapped to PSL objects. These properties can then be assigned properties such as a component word or answer type. Logical axioms are then created to determine which activities must occur at specific times based on activities that have occurred in the past. Axioms
are also created to control which objects participate in the activities and what properties they have.

In the evaluation ontology a user event, such as asking a question, is asserted within the knowledge base as a new occurrence of an activity, such as \textit{askQuestion}. A new object representing the contents of the activity, such as a question with specific properties, is also asserted within the knowledge base. This new object then participates in the occurrence of the activity at a specific time. For example, one question might be asked several times, each at a different time point. Only one copy of the question exists in the knowledge base but it participates in a unique occurrence of the \textit{askQuestion} activity each time it was asked.

The evaluation decision support system will use logical inference to query the knowledge base to determine if an occurrence of the \textit{generateComment} activity must occur at a specific time. The inference engine will examine what activities have occurred up to that time point and determine if the logical axioms that control the occurrence of a \textit{generateComment} activity have been satisfied. The inference engine will also determine which comment objects participate in the occurrence and what their specific properties might be.

5 Taxonomy

The taxonomy of an ontology specifies the concepts and their relationships that are contained within the knowledge model. The generation of a taxonomy is an important first step in the development of an ontology. The taxonomy in Figure 5-1 illustrates the “is-a” relationship, similar to inheritance in object-orientated programming, between PSL concepts and ITS concepts.
A taxonomy also specifies the relationships that exist between the various concepts. Relation functions have a number of input variables and return true or false. For example, the relation `hasComponentWordType(‘1.1.2’, ‘Jargon’)` would return true if the knowledge base knew that the question object ‘1.1.2’ and the component word property ‘Jargon’ associated with it. The following relations are used in the ITS prototype evaluation ontology:

- `hasComponentWord( object, property)` – where object is a question or comment and property is a componentWord.
- `hasProposedClassification(object, property)` – where object is an saElement and property is a classification.
- `hasCorrectClassification(object, property)` – where object is an saElement and property is a classification.
• hasAnswerType(object, property) – where object is a question and property is an answerType.

• saElementHighlighted(object) – where object is an saElement and the SA element has been highlight through either an saElementRevealed activity or a missedVisualSAElement comment.

• Parent(object, object) – where both objects are a question and the first question is the parent of the second.

• Follows(occurrence1, occurrence2) – the relation is true if the occurrence1 is the first occurrence following occurrence2.

• totalBoundaryDuration(saElement, time) – the relation is true if the total time the cursor has spent within the saElement boundary is greater than the given time.

6 Logical Axioms

Axioms, specified in first order logic, provide a formal way of specifying how the concepts and relations behave in the knowledge model. The primary axioms developed for the evaluation ontology pertain to the PSL participates_in axiom. This axiom specifies under what circumstance an object participates in the occurrence of an activity. By specifying the circumstances under which this axiom is true for various comment objects the ontology can capture the desired expert knowledge.

6.1 Questioning Evaluation Axioms

The goal of the questioning technique component of the evaluation ontology is to use the history of questions asked, and the information about each question, to answer the following competency questions:

1. Should a questioning technique evaluation comment be generated?

2. Should a missed follow-up question evaluation comment be generated?

3. Should a questioning efficiency comment be generated?

The following sections provide the circumstances for when these comments should be generated using a formal, first order logic, approach.
6.1.1 Questioning Technique Comment

A questioningTechnique object c, with a componentWordProperty p, participates in an occurrence of generateComment at time point t if and only if:

- a question object q, with componentWordProperty p, participated in an occurrence of askQuestion at time point t.

\[
\forall p, q, t, a_1, o_1 \text{ question}(q) \land \text{occurrence of }(o_1, a_1) \land \text{askQuestion}(a_1) \\
\land \text{participates in}(q, o_1, t) \land \text{componentWordProperty}(q, p) \\
\Rightarrow \exists c, o_2, a_2 \text{ questioningTechnique}(c) \land \text{occurrence of }(o_2, a_2) \\
\land \text{generateComment}(a_2) \land \text{componentWordProperty}(c, p) \\
\land \text{participates in}(c, o_2, t)
\]

\[
\forall c, o_2, a_2, t, p \text{ questioningTechnique}(c) \land \text{occurrence of }(o_2, a_2) \\
\land \text{generateComment}(a_2) \land \text{componentWordProperty}(c, p) \\
\land \text{participates in}(c, o_2, t) \\
\Rightarrow \exists q, a_1, o_1 \text{ question}(q) \land \text{occurrence of }(o_1, a_1) \land \text{askQuestion}(a_1) \\
\land \text{participates in}(q, o_1, t) \land \text{componentWordProperty}(q, p)
\]

6.1.2 Missed Follow-up Question Comment

A missedFollowupQuestion object c participates in an occurrence of generateComment at time point t if and only if:

- A question q₁, with answerType partial, participated in an occurrence of askQuestion at time point t₁
- A question q₂ participated in an occurrence of askQuestion at time point t₂
- No other askQuestion activities occurred between t₁ and t₂
- q₁ is not the parent of q₂

\[
\forall q_1, t_1, a_1, o_1, q_2, o_2, t_2 \text{ question}(q_1) \land \text{question}(q_2) \land \text{occurrence of }(o_1, a_1) \\
\land \text{occurrence of }(o_2, a_1) \land \text{askQuestion}(a_1) \land \text{participates in}(q_1, o_1, t_1) \\
\land \text{participates in}(q_2, o_2, t_2) \land \text{follows}(o_2, o_1) \land \neg \text{parent}(q_1, q_2) \\
\land \text{hasAnswerType}(q_1, \text{Partial}) \\
\Rightarrow \exists c, o_3, a_2 \text{ MissedFollowupQuestion}(c) \land \text{occurrence of }(o_3, a_2) \\
\land \text{generateComment}(a_2) \land \text{participates in}(c, o_3, t_2)
\]
∀c, a_3, a_2, t_2 MissedFollowupQuestion(c) ∧ occurrence_of(o_3, a_2) ∧ generateComment(a_2) ∧ participates_in(c, o_3, t_2)

⇒ ∃q_1, t_1, a_1, o_1, q_2, o_2 question(q_1) ∧ question(q_2) ∧ occurrence(sofar_1, a_1) ∧ occurrence(o_2, a_1) ∧ askQuestion(a_1) ∧ participates_in(q_1, o_1, t_1) ∧ participates_in(q_2, o_2, t_2) ∧ follows(o_2, o_1) ∧ ¬parent(q_1, q_2) ∧ hasAnswerType(q_1, Partial)

The follows relation is used to determine if two occurrence immediately follow each and that no other occurrences occur between them. The relation holds if and only if:

- An occurrence o_1 occurs before an occurrence o_2
- There does not exist an occurrence o_3 such that o_3 occurs between o_1 and o_2

∀o_1, o_2 occurrence_of(o_1) ∧ occurrence_of(o_2) ∧ before(beginof(o_1), beginof(o_2)) ∧ ¬∃o_3 occurrence_of(o_3) ∧ between(beginof(o_1), beginof(o_3), beginof(o_2) ≡ follows(o_2, o_1)

6.1.3 Questioning Efficiency Comment

A questioningEfficiencyComment object c participates in an occurrence of generateComment, at time point t, if and only if:

- Twenty or more questions have been asked
- The ratio of total questions asked to total SA element revealed is greater than 2.0
- A questioning efficiency comment c has not already been generated

∀t, a_1, o_1, a_2, o_2 askQuestion(a_1) ∧ occurrence(o_1, a_1) ∧ (count(o_1) > 20) ∧ saElementRevealed(a_2) ∧ occurrence(o_2, a_2) ∧ (count(o_1)/count(o_2) > 20) ∧ ¬∃a_3, o_3, t_2, c generatComment(a_3) ∧ occurrence(o_3, a_3) ∧ questioningEfficiencyComment(c) ∧ participates_in(c, o_3, t_2) ∧ (t_2 ≤ t)

⇒ ∃t_3, o_4, o_2 occurrence_of(o_4, a_3) ∧ questioningEfficiencyComment(c_2) ∧ participates_in(c_2, o_4, t_3)

∀t_3, o_4, c_2 occurrence_of(o_4, a_3) ∧ questioningEfficiencyComment(c_2) ∧ participates_in(c_2, o_4, t_3)

⇒ ∃t, a_1, o_1, a_2, o_2 askQuestion(a_1) ∧ occurrence(o_1, a_1) ∧ (count(o_1) > 20) ∧ saElementRevealed(a_2) ∧ occurrence(o_2, a_2) ∧ (count(o_1)/count(o_2) > 20) ∧ ¬∃a_3, o_3, t_2, c generatComment(a_3) ∧ occurrence(o_3, a_3) ∧ questioningEfficiencyComment(c) ∧ participates_in(c, o_3, t_2) ∧ (t_2 ≤ t)
6.2 SA Element Evaluation Axioms

The goal of the SA element component of the evaluation ontology is to use the history of interactions with SA elements: including their discovery, classification and mouse-over status, to answer the following competency questions:

1. Should a wrong SA element classification comment be generated?
2. Should a missed visual SA element classification comment be generated?

The following sections provide the circumstances for when these comments should be generated using a formal, first order logic, approach.

6.2.1 Wrong SA Element Classification Comment

A wrongSAElementClassification object c participates in an occurrence of \( \text{generateComment} \), at time point t, if and only if:

- An saElement \( e \) participated in an occurrence of classifySAElement at time point t
- The proposed classification of \( e \) does not match the correct classification of \( e \)

\[
\forall e, t, a_1, o_1, f \ SAElement(e) \land occurrence(a_1, a_1) \land \text{classifySAElement}(a_1) \\
\land \text{participates_in}(e, o_1, t) \\
\land \neg \text{hasCorrectClassification}(e, f) \land \text{hasProposedClassification}(e, f) \models \exists c, a_2, o_2 \land \text{WrongSAElementClassification}(c) \land \text{occurrence}(o_2, a_2) \land \text{generateComment}(a_2) \\
\land \text{participates_in}(c, o_2, t)
\]

\[
\forall c, a_2, o_2 \land \text{WrongSAElementClassification}(c) \land \text{occurrence}(o_2, a_2) \\
\land \text{generateComment}(a_2) \land \text{participates_in}(c, o_2, t) \\
\models \exists e, t, f, a_1, o_1 \ SAElement(e) \land occurrence(o_1, a_1) \\
\land \text{classifySAElement}(a_1) \land \text{participates_in}(e, o_1, t) \\
\land \neg \text{hasCorrectClassification}(e, f) \land \text{hasProposedClassification}(e, f)
\]

6.2.2 Missed Visual SA Element Comment

A missedVisualSAElement object c participates in an occurrence of \( \text{generateComment} \), at time point t, if and only if:

- The total amount of time spent within the boundary of the SA element \( e \) is less than 3.0 seconds.
- \( e \) is not highlighted
\[ \forall t, e \ SAElement(e) \land totalBoundaryDuration(e, t) \geq 3.0 \]
\[ \Rightarrow \exists c, o, a \ MissedVisualSAElement(c) \]
\[ \land occurrence(c, a) \land generateComment(a) \land participates_in(c, o, t) \]
\[ \land \neg prior( saElementHighlighted(e), o ) \]
\[ \land holds(saElementHighlighted(e), o) \]

\[ \forall t, c, o, a \ MissedVisualSAElement(c) \land occurrence(c, a) \land generateComment(a) \]
\[ \land participates_in(c, o, t) \land \neg prior( saElementHighlighted(e), o ) \]
\[ \land holds(saElementHighlighted(e), o) \]
\[ \Rightarrow \exists e \ SAElement(e) \land totalBoundaryDuration(e, t) \geq 3.0 \]

The total amount of time spent within the boundary of an SA element \( e \), \( totalBoundaryDuration \), is calculated by examining the occurrences of \( saElementBoundaryEntered \) and \( saElementBoundaryExited \). The value of \( totalBoundaryDuration \) is calculated by combining:

- The time duration between occurrences of \( saElementBoundaryEntered \) and \( saElementBoundaryExited \) that follow each other.
- The duration between the current time \( t \) and time point \( t_1 \) if:
  - There exists an occurrence \( saElementBoundaryEntered \) at time point \( t_1 \) and there does not exist an occurrence of \( saElementBoundaryEntered \) between \( t_1 \) and the current time.
Chapter 2
Prolog Implementation

7 Inference Engine Selection

The goal of this project was to develop a stand-alone decision support system (DSS) that would react to events from the student and automatically generate comments that could be used to provide instruction to the student. The DSS would use the logical axioms defined in the ontology, combined with an inference engine, to determine if new `generateComment` activities should occur. Events received from the user interface, such as which question was selected, will act as the foundation knowledge for the inference engine.

With the requirement of the DSS to operate autonomously and in real-time Prolog was chosen as the inference engine. SWI-Prolog was selected for the implementation due to its C++ interface. The C++ interface would allow the development of a communications layer that would receive event messages from the user interface and generate evaluation comment messages when appropriate. In this manner the knowledge base could be updated with new assertions based on received events from the user interface.

As a logic inference engine Prolog uses a closed world approach and requires that all logical axioms be in the form of Horn clauses. These requirements led to several restrictions that required changes to the implementation of the first order logical axioms in both the evaluation and foundational PSL ontologies.

7.1 Closed World Restrictions

In inference engines a closed world approach is a fundamental assumption about when a logical axiom should fail. In a closed world a logical axiom fails if the inference engine cannot find any specific solution and is used to significantly increase the speed of calculations. Due to this a closed world approach restricts the use of the existential quantifier in logical axioms.

The closed world approach in Prolog limits the use of existential quantifiers, $\exists$, axioms to infer the existence of new comments or occurrences in the knowledge base. In the Prolog implementation of the evaluation ontology the solution to this problem was to have a single comment object for each type and a single occurrence of `generateComment`. The
generateComment occurrence would then participate with the comment object of a specific type at multiple different times. This approach also made it necessary for properties of the specific comment object to change over time.

7.2 PSL in Prolog

The axioms within the PSL ontology are written in first-order logic and cannot be translated into Horn clauses that are compatible Prolog. However, as many of these axioms act as integrity constraints it was possible to use the PSL as a guiding philosophy of the evaluation ontology. Implementing the PSL within Prolog was also simplified as the queries to the inference engine did not require most of the axioms from the PSL core ontology.

The primary axioms implemented from the PSL where those dealing time-points and durations. In the Prolog implementation time-points were restricted to real numbers larger than zero. No concept of infinite time was required which allowed arithmetic comparison to be used to evaluate the before axioms. The sub-class axioms where also implemented, however the closed world approach of Prolog limited their use to integrity constraints.

8 Axiom Conversion

To implement the evaluation ontology in a Prolog inference engine the first order axioms specified in Section 6 must be converted to Horn clauses. A Prolog Horn clause, or rule, specifies the head, or goal, on the left separated from the body, or restrictions, by a “:-“ symbol. Prolog iterates through all instances of objects in its knowledge base in order to satisfy the restrictions of the stated query, either through direct assertions in the knowledge base of by satisfying sub-goals.

The process of converting the first order axioms in the evaluation ontology into Prolog goals that could be used to solve queries such as what types of comments should participate in the generateComment activity at a specific time required an understanding of how to select the desired object instances in the knowledge base and restrict their ability to satisfy the goals accordingly. The following sections describe the specific Prolog goals for each axiom and how they meet this requirement.
8.1 Questioning Evaluation Axioms

The following sections present Prolog statements, adapted from the formal logical axioms presented Section 6.1. These statements are used in the inference engine to answer the following questioning technique competency questions of the ontology:

1. Should a questioning technique comment be generated?
2. Should a missed follow-up question comment be generated?
3. Should a questioning efficiency comment be generated?

8.1.1.1 Questioning Technique Comment

To satisfy the goal of having a questioningTechnique comment, Object, participate in a generateComment activity, Occurrence, at a specific time point, Time, Prolog must be able to find the following information in the knowledge base:

- A question object, Question, exists and it does not have the componentWord property ‘None’ at Time.
- The question object, Question, participated in the askQuestion activity at Time.

\[
\text{participates_in(Object, Occurrence, Time) :- questioningTechnique(Object),}
\]
\[
generateComment(Activity),
\]
\[
occurrence_of(Occurrence, Activity),
\]
\[
question(Question),
\]
\[
\neg \text{hasComponentWord(Question, 'None', Time),}
\]
\[
askQuestion(Activity2),
\]
\[
occurrence_of(Occurrence2, Activity2),
\]
\[
\text{participates_in(Question, Occurrence2, Time).}
\]

8.1.1.2 Missed Follow-up Question Comment

To satisfy the goal of having a missedFollowupQuestion comment, Object, participate in a generateComment activity, Occurrence, at a specific time point, Time, Prolog must be able to find the following information in the knowledge base:

- Two unique question objects, Q and Q2 exist.
- Q participates in a questionActivity, QOcc at time point Time.
- Q2 participates in a questionActivity, QOcc2 at time point T1.
- The asking of Q2 at T1 was the nearest question asked to Q1 at Time, i.e. the asking of Q2 follows the asking of Q1.
- Q2 has an answerType property of ‘PARTIAL_ANSWER’
- Q2 is not the parent of Q1 within the dialog tree.
participates_in(Object, Occurrence, Time) :- missedFollowupQuestion(Object),
generateComment(Activity),
occurrence_of(Occurrence, Activity),
occurrence_of(QOcc, questionActivity),
question(Q),
participates_in(Q, QOcc, Time),
question(Q2),
Q \= Q2,
hasAnswerType(Q2,'PARTIAL_ANSWER'),
occurrence_of(QOcc2, questionActivity),
participates_in(Q2, QOcc2, T1),
\+ parent(Q2,Q),
nearestQuestionAskedTimePoint(Q2, T1, Time).

Determining the previously asked question should require the use of the follows axiom specified Section 6.1.2 however the closed world approach within Prolog requires a different approach. To query this information in Prolog the nearestQuestionAskedTimePoint goal is used to determine the question object and at what time it was asked.

This goal uses the findall query to determine the set of times, before the reference time, where a question was asked. The largest value of this set is used to obtain the question object that participated at that time.

nearestQuestionAskedTimePoint(Question, T1, Time) :-  question(Question),
occurrence_of(Occurrence, questionActivity),
findall(X, ( question(Q),
participates_in(Q, Occ, X),
occurrence_of(Occ, questionActivity),
before(X, Time)), Set),
max_list(Set,T1),
participates_in(Question,Occurrence,T1).

8.1.2 Questioning Efficiency Comment

To satisfy the goal of having a questioningEfficiency comment, Object, participate in a generateComment activity, Occurrence, at time point, Time, Prolog must be able to find the following information in the knowledge base:

- The question efficiency limit was reached at Time.
- The previous occurrence of askQuestion at time T1 did not trigger the limit, ensuring the comment is only generated once.

participates_in(Object, Occurrence, Time) :- questioningEfficiency(Object),
generateComment(Activity),
occurrence_of(Occurrence, Activity),
questionEfficiencyLimitReached(Time),
nearestQuestionAskedTimePoint(Q, T1, Time),
\+ questionEfficiencyLimitReached(T1).
To determine if the question efficiency limit is reached Prolog must determine if the number of questions asked up to Time is greater than 20 and that the ratio of SA elements revealed to questions is less than 0.5. To determine the number of questions asked or SA elements revealed uses the findall query to find the list of all question objects that participate in a questionActivity occurrence before the current time point, Time.

\[
\text{numberOfQuestionsAsked( Time, Num):- findall(X, ( occurrence_of(X, questionActivity), question(X1), participates_in(X1, X, T), beforeEq(T, Time), Set1), length(Set1, Y), Num is Y).}
\]

8.2 SA Element Evaluation Axioms

The following sections present Prolog statements, adapted from the formal logical axioms presented Section 6.2. These statements are used in the inference engine to answer the following SA element competency questions of the ontology:

1. Should a wrong SA element classification comment be generated?

2. Should a missed visual SA element comment be generated?

8.3 Wrong SA Element Classification Comment

To satisfy the goal of having a wrongSAElementClassification comment, Object, participate in a generateComment activity, Occurrence, at time point, Time, Prolog must be able to find the following information in the knowledge base:

- Element was not correctlyClassified

\[
\text{participates_in(Object, Occurrence, Time) :- wrongSAElementClassification(Object), generateComment(Activity), occurrence_of(Occurrence, Activity), saElement(Element), classifySAElement(Activity2), occurrence_of(Occurrence2, Activity2), participates_in(Element, Occurrence2, Time), \neg correctlyClassified(Element).}
\]

To determine if an SA element was correctly classified Prolog compares the proposed classification to the correct classification. Both classifications are entered into the knowledge
base by the communications layer of the Evaluation Module upon receipt of a classification event.

A classification, which may be a combination of timer, victim or command, is entered into the knowledge base as a set containing the terms. The correctlyClassified goal uses the built-in sort function to ensure the lists are in the same order and determines if they are equal.

\[
\text{correctlyClassified}(SAElement) :- \text{hasCorrectClassification}(SAElement,CorrectList), \text{hasProposedClassification}(SAElement,ProposedList), \text{sort}(CorrectList,X), \text{sort}(ProposedList,Y), X = Y.
\]

### 8.3.1 Missed Visual SA Element Comment

To satisfy the goal of having a missedVisualSAElement comment, Object, participate in a generateComment activity, Occurrence, at a specific time point, Time, Prolog must be able to find the following information in the knowledge base:

- An \textit{saElement}, Element which has not already been activated by Time.
- The boundaryDuration, Value, is greater than 3.0 seconds.

\[
\text{participates_in}(Object, Occurrence, Time) :- \text{missedVisualSAElement}(Object), \text{generateComment}(Activity), \text{occurrence_of}(Occurrence, Activity), \text{saElement}(Element), \text{totalBoundaryDuration}(Element,Time, Value), Value >= 3.0, \neg \text{activated}(Element, Time).
\]

To determine if an \textit{saElement}, Element, is activated at a time point, Time, Prolog determines if either:

- Element was highlighted at a time point, T1, through the communications layer of the Evaluation Module asserting the event.
- An \textit{saElementRevealed} activity, Occurrence2, participated with the element at a timepoint T1.
- T1 occurred before Time.

\[
\text{activated}(Element, Time) :- \text{saElement}(Element), \text{timepoint}(Time), ( \text{saElementHighlighted}(Element, T1) ; (\text{participates_in}(Element, Occ, Time), \text{occurrence_of}(Occ,\text{saRevealedActivity})) ), \text{before}(T1,Time).
\]
To determine the `totalBoundaryDuration` for an `saElement`, `Element`, at a specific time point, `Time`, Prolog must sum the durations between previous `boundaryEntered` and `boundaryExited` activity occurrences. Prolog must also handle several boundary cases, such as if no `boundaryExited` activity exists.

The first step is to find the sum of the intermediate boundaries between all `boundaryEntered` and `boundaryExited`. The sum of these boundaries, Temp, is then added to duration of the current boundary interval, that is the duration between the current time point, `Time` and the previous `boundaryEntered` activity occurrence.

```
 totalBoundaryDuration(Element, Time, Value) :- findall(Z, intermediateBoundaryDuration(Element, Z), Set),
                 sumlist(Set,Temp),
                 currentBoundaryDuration(Element, Time, Temp2),
                 Value is Temp + Temp2.
```

Each `intermediateBoundaryDuration` is calculated by finding the `intermediateBoundaryInterval` between two time points for an interval. This is done by finding occurrences of the `boundaryEntered` and the `boundaryExited` that follow each other. A similar method to implementing the follows axiom is used in `nearestEnteredTimePoint`.

```
 intermediateBoundaryInterval(Element, Start, End) :- saElement(Element),
                  participates_in(Element, Occurrence, Start),
                  occurrence_of(Occurrence, boundaryEnteredActivity),
                  participates_in(Element, Occurrence2, End),
                  occurrence_of(Occurrence2, boundaryExitedActivity),
                  before(Start,End),
                  nearestEnteredTimePoint(Element, Start, End).
```

Finally the `currentBoundaryDuration` is determined by finding the nearest occurrence of the `boundaryEntered` and `boundaryExited` activities such that the `boundaryExited` activity occurred before the `boundaryEntered` activity, indicated the cursor is still within the SA element boundary.

```
 currentBoundaryDuration(Element, Time, Value) :- saElement(Element),
                  participates_in(Element, Occurrence, T1),
                  occurrence_of(Occurrence, boundaryEnteredActivity),
                  participates_in(Element, Occurrence2, T2),
                  occurrence_of(Occurrence2, boundaryExitedActivity),
                  before(T1,Time),
                  before(T2,Time),
                  before(T2,T1),
                  nearestEnteredTimePoint(Element, T1, Time),
```
8.4 Inference Technique

The role of the decision support system is to automatically generate evaluation comments based on user events such as which questions were asked. To do this the DSS implements the axioms of the evaluation ontology in Prolog as described above. New question or SA element objects are asserted in the Prolog knowledge base as well as the associated occurrences and participates-in relationships. The DSS then periodically performs a Prolog query to determine if any comment objects satisfies the participates_in relationship at that specific time, T. The exact Prolog query is:

\[
\text{participates\_in}(X,Y,T), \text{comment}(X), \text{occurrence\_of}(Y,Z), \text{generateComment}(Z).
\]

The Prolog inference engine will then determine if one of the participates_in goal statements listed above is satisfied. If a solution is returned the DSS determines what type of comment object satisfies the variable X and generate the corresponding evaluation comment message in the communications layer.
Chapter 3
Evaluation

9  First Order Logic Verification

The first stage in evaluating an ontology is the verification of the logical axioms. Logical axioms should be evaluated to ensure they are consistent and satisfiable. A set of logical axioms are consistent if it is impossible to find a contradiction within the axioms and is satisfiable if it is possible to find a situation under which all the axioms are true. A common tool to use to evaluate consistency and satisfiability in first order logic is Prover9. Prover9 also the user to input axioms and then use those axioms to determine if a goal is reachable. If a set of logical axioms is inconsistent these goals may not be reachable within Prover9. An set of axioms can be satisfiable if an intended model can be constructed that uses the ontology axioms to meet a goal. The intended models discussed below and the structure of the evaluation ontology allowed Prover9 to be used to determine if the ontology is satisfiable.

9.1  Intended Models

To verify the ability of the evaluation ontology to supported the desired operation of the decision support system it was necessary to develop the intended models of the ontology and verify that they were supported by the axioms when implemented within Prover9. An intended model is the set of variable assignments within the axioms that represents the desired behaviour of the ontology. In the case of the evaluation ontology proposed in this paper the intended models are the situations under which comments should be generated and consists of the set of input activities occurrences, such as askQuestion, saElementRevealed and classifySAElement along with the corresponding objects that should be present to trigger the desired generateComment activity occurrences. Chapter 12 in the Appendix presents an intended model of the DSS inputs that should generated the desired comment outputs.

9.2  Prover9 Implementation

To verify the intended models within Prover9 the ontology axioms were implemented within the reasoned and the tested in two stages. The first stage of verification was to evaluate the
supporting axioms to ensure they provided the desired foundation within the intended models. The second stage was to evaluate the main axioms against the intended models.

Prover9 supports the use of first order logic axioms within its reasoner, simplify the process of implementing the evaluation ontology axioms. However it is challenging to implement mathematical components of axioms within Prover9 which limited the ability to verify the entire evaluation ontology. As such, only the axioms that deal with the generation of questioning technique, missed follow-up question and wrong SA element classification comments were evaluated within Prover9.

The goal of the Prover9 verification is then to implement the ontology axioms and define the input intended model. A logical goal statement is then created that matches the desired output intended model. If Prover9 can find a successful path to infer the desired output goal from the axioms and input model then the ontology is verifiable and satisfiable with respect to the intended model. The following logical sentences were used to verify the desired behaviour of the intended models:

1. Given an occurrence of a *questionAsked* activity, at time $T$, which has the question object $q$ with component property $p$:

   $$\exists c, o_2, a_2 \text{questioningTechniqueComment}(c) \land \text{occurrence_of}(o_2, a_2) \land \text{generateComment}(a_2) \land \text{componentWordProperty}(c, p) \land \text{participates_in}(c, o_2, T)$$

2. Given the occurrence of two *questionAsked* activities, at time $T_1$ and $T_2$, which have the question objects $q_1$ and $q_2$ and $\neg$parent($q_1, q_2$):

   $$\exists c, o_2, a_2 \text{missedFollowupQuestionComment}(c) \land \text{occurrence_of}(o_2, a_2) \land \text{generateComment}(a_2) \land \text{participates_in}(c, o_2, T_2)$$

3. Given the occurrence of a *classifySAElement* activity, at time $T$, which has the SA element $e$ and the proposedClassification($e$) does not match the correctClassification($e$):

   $$\exists c, o_2, a_2 \text{wrongSAElementClassification}(c) \land \text{occurrence_of}(o_2, a_2) \land \text{generateComment}(a_2) \land \text{participates_in}(c, o_2, T_2)$$
9.2.1 Follows Axiom

The follows axiom is used to specify the linear order of activity occurrences. It allows the decision support system to ensure that a specific activity occurrence directly follows another.

The axiom was entered into Prover9 as:

\[
\begin{align*}
\text{all } x \text{ all } y \\
& \quad ( \text{before}(\text{beginof}(x), \text{beginof}(y)) \& \text{activity occurrence}(x) \& \text{activity occurrence}(y) \& \\
& \quad \neg (\exists z \ (\text{activity occurrence}(z) \& \text{between}(\text{beginof}(x), \text{beginof}(z), \text{beginof}(y)))) \\
& \quad \leftrightarrow \text{follows}(y, x) )
\end{align*}
\]

The intended model specified in Table A-1 of the Appendix would have the following goals, with regards to the follows axiom:

\[
\begin{align*}
\text{follows}(occ2, occ1). & \quad \text{follows}(occ3, occ2). \\
\text{follows}(occ4, occ3). & \quad \text{follows}(occ5, occ4).
\end{align*}
\]

For Prover9 to return a successful proof of these goals an further axiom is needed to close the world and ensure the

\[
\neg \exists \text{occurrence of } (o_3) \land \text{between}(\text{beginof}(o_1), \text{beginof}(o_2), \text{beginof}(o_2))
\]

axiom is successful:

\[
\begin{align*}
\text{all } x \ (\text{activity occurrence}(x) \\
& \quad \leftrightarrow \ ( (x=occ1) \mid (x=occ2) \mid \\
& \quad (x=occ3) \mid (x=occ4) \mid \\
& \quad (x=occ5) )).
\end{align*}
\]

To ensure that no unintended models occur it is necessary to ensure Prover9 cannot prove anything inconsistent with the intended model and that it can prove that a statement is not true in the model. For example Prover9 is unable to prove the following goal:

\[
\text{follows}(occ2, occ3).
\]

Which is not an intended model of the ontology, however Prover9 is able to prove that this axiom is not allowed by the ontology, as desired by the ontology.

\[
\neg \text{follows}(occ2, occ3).
\]
9.2.2 Questioning Technique Comment

The axiom that dictates the existence of a questioning technique comment was implemented in Prover9 in the following manner:

\[
\begin{align*}
\forall p \forall q \forall t \forall a1 \forall o1 \forall a2 \forall o2 \\
& \left( \text{question}(q) \land \text{occurrence_of}(o1,a1) \land \text{askQuestion}(a1) \land \text{participates_in}(q,o1,t) \land \text{hasComponentWordProperty}(q,p) \right) \\
& \implies \left( \exists c \exists o2 \exists a2 \\
& \quad \left( \text{questioningTechniqueComment}(c) \land \text{occurrence_of}(o2,a2) \land \text{generateComment}(a2) \land \text{hasComponentWordProperty}(c,p) \land \\
& \quad \text{participates_in}(c,o2,t) \right) \right).
\end{align*}
\]

\[
\begin{align*}
\forall c \forall o2 \forall a2 \forall p \\
& \left( \text{questioningTechniqueComment}(c) \land \text{occurrence_of}(o2,a2) \land \text{generateComment}(a2) \land \text{hasComponentWordProperty}(c,p) \land \text{participates_in}(c,o2,t) \right) \\
& \implies \left( \exists q \exists o1 \exists a1 \\
& \quad \left( \text{question}(q) \land \text{occurrence_of}(o1,a1) \land \text{askQuestion}(a1) \land \text{participates_in}(q,o1,t) \land \text{hasComponentWordProperty}(q,p) \right) \right).
\end{align*}
\]

Given the history of questions asked in Table A-1 of the Appendix the intended model of the ontology is the occurrence of specific comments given in Table A-2 of the Appendix. The validity of the intended model within the ontology axioms was confirmed in Prover9 through successful proofs of the following goal:

\[
\exists c \exists o \\
\quad \left( \text{questioningTechniqueComment}(c) \land \text{participates_in}(c,o,t1) \land \text{hasComponentWordProperty}(c,\text{None}) \right).
\]

Given the open world nature of Prover9 it is not possible to find a confirming or denying proof, using the current axioms, for goals of the following type:

\[
\begin{align*}
\exists c \exists o \\
& \quad \left( \text{questioningTechniqueComment}(c) \land \text{participates_in}(c,o,t1) \land \text{hasComponentWordProperty}(c,\text{Biased}) \right). \\
\neg \left( \exists c \exists o \\
& \quad \left( \text{questioningTechniqueComment}(c) \land \text{participates_in}(c,o,t1) \land \text{hasComponentWordProperty}(c,\text{Biased}) \right) \right).
\end{align*}
\]

In the intended model, the top goal should not provide a proof, as no question with a componentWord(Biased) was asked at t1, while a proof for its negation should succeed.

9.2.3 Missed Follow-Up Question Comment

The axiom that dictates the existence of a missed follow-up question comment was implemented in Prover 9 in the following manner:

\[
\begin{align*}
\forall q1 \forall t1 \forall a1 \forall o1 \forall q2 \forall o2 \forall t2 \\
& \left( \text{question}(q1) \land \text{occurrence_of}(o1,a1) \land \text{askQuestion}(a1) \land \text{participates_in}(q1,o1,t1) \land \text{hasComponentWordProperty}(q1,p) \right) \\
& \implies \left( \exists c \exists o2 \exists a2 \\
& \quad \left( \text{questioningTechniqueComment}(c) \land \text{occurrence_of}(o2,a2) \land \text{generateComment}(a2) \land \text{hasComponentWordProperty}(c,p) \land \\
& \quad \text{participates_in}(c,o2,t1) \right) \right).
\end{align*}
\]
Due to the open world nature of Prover 9 it is not possible to use the following goal to confirm the existence of a missedFollowupQuestionComment as stated in our intended model.

$$\exists c \exists o \ (\text{missedFollowupQuestionComment}(c) \land \text{participates_in}(c,o,t3)).$$

The reason for this is due to the dependency of this goal on the proving of $\text{follows(qOcc3,qOcc2)}$. As described in the Supporting Axioms section the ability to prove the follows axiom requires the introduction of axioms to close the world with regards to activity_occurrences. Under this circumstance it becomes difficult to use Prover 9 to confirm the existence of an occurrence of such a comment.

### 9.2.4 Wrong SA Element Classification Comment

The axiom that dictates the existence of a wrong SA element classification comment was implemented in Prover9 in the following manner:

$$\forall c \forall e \forall t \forall a1 \forall o1 \forall o2 \forall a2 \forall q1 \forall q2 \ (\text{saElement}(e) \land \text{occurrence_of}(o1,a1) \land \text{classifySAElement}(a1) \land \text{participates_in}(e,o1,t) \land \text{hasProposedClassification}(c,e) \land \neg \text{haCorrectClassification}(c,e) \land \text{follows}(o2,o1) \land \neg \text{parent}(q1,q2) \land \text{hasAnswerType}(q1,\text{Partial}) \land \exists c \exists o2 \exists a2 \ (\text{wrongSAElementClassification}(c) \land \text{occurrence_of}(o2,a2) \land \text{generateComment}(a2) \land \text{participates_in}(c,o2,t))).$$
Given the history of SA element classification activities presented in Table A-1 of the Appendix the intended model of the ontology is the occurrence of specific comments given in Table A-2 of the Appendix. The validity of the intended model for SA element classification comments would be confirmed through the successful proofs of the following goals:

\[
\exists c \exists o \ ( \text{wrongSAElementClassification}(c) \land \text{participates_in}(c,o,t6) ).
\]

However, the open world nature of Prover9 makes it difficult to find a proof that satisfies the requirement that the \text{hasProposedClassification} and \text{hasCorrectClassification} are not equivalent. Although a closed world around the \text{saElement} and \text{classification} objects was implemented Prover9 continued to be unable to provide a proof.

### 9.3 Prover9 Limitations

Prover9 represents a powerful method for evaluating the completeness of a set of logical axioms, however the power of its open world methodology has shown it to be difficult to use in the verification of the evaluation ontology presented in this report. In an open world methodology the reasoner will continue to search for possible instances that satisfy an axiom and will not return false if it cannot prove something. The opposite of this approach is a closed world methodology where the reasoner will search its known space and, if it cannot find a specific satisfaction case, will return false. The inability of Prover9 to successfully find proofs for goals in the simple intended models provided show that a much more complicated set of logical axioms, including ones that define the entire closed world of possibilities will be required to satisfy the necessary proofs.

Prover9 also provides a limited ability to perform math calculations. This limitation prevented the implementation of the questioning efficiency comment axiom and the missed visual SA element comment axiom in the reasoner.

The original intention of the evaluation ontology is for it to be used within a decision support system that would operate in real-time and generate comments from events generated in a user interface. The open world limitations of Prover9 indicate that the closed world reasoner within Prolog is the appropriate tool for the DSS. The following chapters describes how a Prolog
inference engine, using the evaluation ontology, was integrated into a prototype DSS and used to both verify the ontology and to evaluate its performance.

10 Decision Support System Prototype

To verify the ontology for its practical use and to evaluate the performance of the Prolog implementation of the evaluation ontology software prototypes of the decision support system and associated tutoring system were developed. A user interface to the tutoring system was developed that allowed the student to select a question to ask from the dialog tree discussed in section 2.1. The student could also interact with SA elements and where appropriate provide a proposed classification.

These events from the user were transmitted to the inference engine using a UDP network communication protocol. The interference engine used SWI-Prolog to implement the evaluation ontology and perform frequent queries to determine if new comments should be generated and sent to the student interface through the communications layer.

Finally the complete decision support system to evaluate the students behaviour was tested to ensure the desired comments were generated at the correct point and in a timely manner.

10.1 Student Interface

The student interface allows the user to interact with the scenario, selecting questions to ask an NPC and investigating the visual components of the IED. The graphical user interface (GUI) is based on the Windows Presentation Foundation (WPF) API and consists of three main components:

- Question Widget – allows the student to select a question to ask the NPC and navigate down the dialog tree.
- Answer Widget – displays the answers to any questions asked and any evaluation feedback that is received from the inference engine.
- Scene Widget – displays the visual scene and allows the user to select NPCs to question and IEDs to investigate.
The student begins the scenario by selecting an NPC to question. The top-level questions are presented in the Question Widget and as the student selects questions dialog interactions are sent to the evaluation engine. When the inference engine generates a comment based on the evaluation ontology an instructional feedback is presented to the student in the answer widget. When the student is finished with questioning the NPC they may interact with the scene widget to investigate the IEDs, events from these investigations are sent to the inference engine for evaluation.

10.1.1 Question Widget

The question widget displays a selection of questions from the dialog tree. The selection represents the list of questions the student has direct access, starting with the top-level root questions. As the student selects a question, the list of follow-up questions is presented in the next column.

The question list is extract from an XML file which contains both the question and answer text as well as the properties required for the evaluation ontology: the unique identification of the question, the answer type and the component word property. When a question is selected the answer text is displayed in the answer widget and an event is transmitted to the inference engine containing the specific question ID, answer type and component word property.

A question may also have a list of SA elements that it reveals when the question is asked. This list is also defined in the dialog tree. When the student asks the question an SA element revealed event is sent to the inference engine. The student is then asked to classify the SA element, which also sends an event containing the proposed and correct classifications to the inference engine. If the student classifies the SA element incorrectly an evaluation comment is generated in the inference engine and instructional feedback is presented to the student.
10.1.2 Answer Widget

The answer widget displays the history of questions asked and answers received. It also presents evaluation feedback received from the inference engine. When the student clicks the instructor feedback button, instructional content appropriate to the evaluation comment is presented.

10-2 Answer Widget

10-3 Evaluation Feedback

The instructional feedback displayed in answer widget and the instructor dialog are the primary way for displaying the result of the inference engine and were used to verify the behaviour of the evaluation ontology in prolonged test scenarios.

10.1.3 Scene Widget

The scene widget allows the student to interact with the scenario. The student can select an NPC to question or navigate to an IED and investigate it for visual SA elements. When the student
clicks on a witness NPC in the reception screen, the dialog is initiated and the dialog tree for that NPC is loaded into the question widget.

10-4 Scene Widget "Reception"

The scene widget is also able to present active images as a means to investigate the visual components of the IEDs in the scenario. The visual SA elements are defined in the scenario and connected to images on the screen. In the backpack device outside the office, shown below, the wires, battery pack and timer are all visual SA elements.
When the student clicks on any of these images in the scene widget an SA element revealed event is sent to the inference engine and the student is asked to classify the device. The classification is evaluated in the inference engine and, if appropriate, instructional feedback is presented.

10.2 Inference Engine Implementation

The inference engine implements the Prolog evaluation ontology, documented in Section 8, using the C++ interface to SWI-Prolog. The inference engine consists of three parts; a communications layer that processes events from the user interface, the Prolog inference engine, and a management layer that converts between events and the Prolog knowledge base.

10.2.1 SWI Prolog Engine

The SWI-Prolog Engine uses the existing Prolog application that defines the evaluation ontology but provides a simplified interface for asserting new relation information into the knowledge base, such as the occurrence and associated details of an askQuestion activity. The engine also provides an interface for querying new knowledge that is created through the use of the evaluation ontology inference rules.
To assert new relations in the Prolog knowledge base the relation must first be defined as dynamic in the application rules. Then the SWI-Prolog C++ interface is used to programmatically execute the equivalent of the Prolog rule `assert(relation(param1,param2..))`. As new events are received from the user interface the management layer is responsible for interpreting the information and using the SWI-Prolog Engine to assert the required relations within the knowledge base.

The SWI-Prolog Engine can also be used to query the existence of information in the ontology knowledge base by programmatically querying the Prolog application. The user provides the relation name and the set of parameters to query. As in using the stand-alone Prolog application a number of variables can be used as the parameters. The query will then return the instances of each parameter that satisfy the relation. To determine if the comment objects participate in new occurrences of the `generateComment` activity the management layer will periodically query the `participates_in` relation and examine the resulting instances.

### 10.2.2 Management Layer

The inference engine management layer is responsible for converting the information contained within events received from the user interface, such as which question has been asked, into evaluation ontology specific information within the Pro-Log knowledge base. The management layer is also responsible for querying the evaluation ontology regarding the generation of evaluation comments and, when needed, sending these comments to the user interface for presentation to the student. The management layer is divided into two components, the questioning technique evaluation and the SA element evaluation.

#### 10.2.2.1 Questioning Technique Evaluation

The questioning technique evaluation component of the management layer is responsible for handling all dialog interaction events from the user interface. When the student selects a question the specific information, such as component word property and question ID, is packaged into a communication event. The questioning technique evaluation component receives that event and performs the following activities:

1. Determines if the evaluation ontology has knowledge about the specific question.
a. If no information exists, the management layer asserts the existence of the question and the various relations, such as component word property.

2. Generates a new occurrence of questionActivity and asserts the participates_in relation between the question and occurrence at the current time.

Following this the questioning technique evaluation component queries the SWI-Prolog knowledge base to determine if any occurrences of commentActivity satisfy the relation for participates_in at the current time. If any of the evaluation ontology rules for generating comments have been met this query will return positive information. This information is used to generate an evaluation comment that is sent to the user interface.

10.2.2.2 SA Element Evaluation

The SA element evaluation component of the management layer is responsible for handling all SA element related events. These events include classification of elements and the entering or exiting of boundaries. When the student performs the appropriate action involving an SA element the appropriate information is packaged into a communications event. The SA element evaluation component receives this event and performs the following activities:

1. Determines if the evaluation ontology has information regarding the specific SA element involved

   a. If no information exists, the management layer asserts the existence of the SA element and the various relations, such as the correct classification.

2. Generates a new occurrence of the specific SA element activity, such as saElementBoundaryEntered, and asserts the participates_in relation between the SA element and occurrence at the current time.

Periodically the questioning technique evaluation comment will query the SWI-Prolog knowledge base to determine if any occurrence of commentActivity satisfy the relation participates_in at the current time. If any of the evaluation ontology rules, specific to SA elements, for generating comments have been met this query will return positive information. This information is used to generate an evaluation comment that is sent to the user interface.
10.3 Verification and Evaluation

Two phases of evaluation and verification testing was performed on the DSS prototype. During its development structured testing was performed to ensure the correct behaviour of the evaluation ontology and the inference engine implementation. This testing involved the use of carefully structured scenarios to test all aspects of evaluation ontology. When the DSS prototype was confirmed to have satisfy the requirements of the evaluation ontology a large scenario was developed to evaluate the performance of the inference engine under real-time loads.

10.3.1 Functional Verification

The functional testing of the DSS prototype consisted two stages: ensuring the functionality of the evaluation ontology within Prolog and testing the implementation of the inference engine in coordination with the user interface. To verify the functionality of the evaluation ontology within Prolog several stub programs were written that populated the Prolog knowledge base with the information on question and SA element activities. This information was designed to ensure the various evaluation comments within the ontology would be generated. The resulting Prolog applications were then run and queries executed to ensure the evaluation comment objects satisfy the \textit{participates in} relation at the correct times.

Once the Prolog implementation of the evaluation ontology was verified the inference engine was implemented along with the communications layer to connect it to the user interface. A simplified version of the user interface scenario, which consists of the questioning information and the SA element information, was then used to ensure the correct behaviour of the inference engine. The inference engine testing consisted of designing the user interface scenario so that the evaluation ontology comments could be efficiently activated. Once the behaviour of the inference engine was verified to be consist with the behaviour of the stand-alone Prolog ontology and that all evaluation comments could be generated using the user interface the performance of the interference engine in a real-time environment was evaluated.

10.3.2 Performance Evaluation

To evaluate the performance of the inference engine and evaluation ontology in a real-time environment a large scenario for the user interface was developed. This scenario consisted of several visual SA elements and a large number of questions. The details of the questions were
distributed such that numerous evaluation comments would be generated, but that there was also a large number of questions that would not generate any comments. The goal of this testing was to ensure the inference would meet the requirements of the DSS prototype and be able to generate comments in a real-time environment.

The results of the testing should that the inference engine would slow down significantly over time when a large amount of information was in the knowledge base. The large number of instances that Prolog would need to iterate through to satisfy the evaluation ontology relations was increasing the amount of time a call to query the knowledge base would take. The extended time would over run the time slice assigned to the inference engine and would significantly delay the generation of an evaluation comment from an event in the user interface. This performance problem was found to occur in both circumstances when a large number of questions had been asked and when mouse entered and exited the boundary of SA elements frequently. Several modifications were necessary to the implementation of both the evaluation ontology rules in Prolog and the inference engine.

10.3.2.1 Modifications

To improve the performance of the inference engine to handle large amounts of questions and SA element information modifications where needed to the evaluation ontology implementation in Prolog as well as a change to how the inference engine operated.

The Prolog implementation of the evaluation ontology rules was done such that Prolog would perform more queries than was strictly necessary. An example of this was in the missed-follow up question comment generation axiom where Prolog was used to determine the which occurrences of questionActivity followed each other. This proved to be a significant cause of the performance issues and a modification was made such that the inference engine specified the follows relation when the dialog events where received.

The second area for improvements in the Prolog implementation was the need for the inference engine to determine properties about the evaluation comments after they had been generated. Additional Prolog rules were required to link the evaluation comments with the appropriate details, usually as a result of the question that triggered the comment. Instead of these additional
rules the \textit{participates\_in} rule that generated the comment was modified such that the \textit{assert} relation was called to establish the link between the triggering property and the comment.

Finally the inference engine was originally designed to periodically query the Prolog knowledge base, even when an event had not occurred. This behaviour was a result of the need to generate the \textit{missedVisualSAElement} comment when the students mouse was within the boundary, not when the mouse exited the boundary and a corresponding event was received. The \textit{missedVisualSAElement} comment requirements were modified such that the comment would only be generated when the mouse entered or exited the visual SA element boundary, easing the load on the inference engine.
11 Ontology Summary

The goal of the IED evaluation ontology presented in this report is to generate evaluation comments based on student interactions. The types of comments that can be generated are summarized in the competency questions defined in Section 3. The competency questions related to asking a question are:

- Should a questioning technique comment be generated?
- Should a missed follow-up question comment be generated?
- Should a questioning efficiency comment be generated?

The competency questions related to revealing and evaluating SA elements are:

- Should a wrong SA element classification comment be generated?
- Should a missed visual SA element comment be generated?

Using the PSL as a foundational ontology the taxonomy, presented in Section 4, outlines what types of activities and objects exists. The following student interaction activities are used as inputs into the competency questions:

- Asking a question,
- Revealing, classifying and entering or exiting the boundary of an SA element.

The following objects and properties are associated with the activities in the generation of comments:

- The answer type and component word properties of a question,
- The correct and proposed classification of an SA element.
The first-order logic axioms, presented in Section 6, and their implementation in Prolog rules, presented in Section 8, are used to control the circumstances under which the associated competency question is answered and an evaluation comment is generated. The axioms restrict the PSL `participates_in` foundation axiom and map the occurrence of an evaluation comment to a set of restrictions based on the input activities. Each axiom is presented in its own sub-section and maps directly to a competency question.

Evaluating the first-order logical axioms proved challenging in Prover9 due to the open world nature of the tool. However the implementation of the evaluation ontology into a real-time, Prolog based, inference engine demonstrated the viability of the axioms and met the requirements posed by the competency questions.

12 Conclusion

To correctly dispose an IED required the trained operator gather information about the device and situation in order to determine the IED trigger type. A key task in gathering this information is the questioning of witnesses to obtain information elements about the scenario. The goal of this project was to develop an ontology that can be used to evaluate a student’s questioning technique and generate comments that can be used to provide instructional feedback. The ontology was then integrated into a decision support system within a prototype tutoring system. The DSS would monitor the questions a student had selected and generate evaluation comments at the appropriate time.

The Process Specification Language was chosen as the foundational ontology because the evaluation ontology involved the asking of questions, interacting with SA elements and generating of comments, which mapped well to concepts of objects participating in the occurrences of activities. In the final taxonomy of the ontology asking questions, interacting with SA elements and generating comments where activities while individual comments, questions and SA elements where objects that had specific properties. Axioms where then generated that related the existence of comments to the participation of questions or SA elements, with specific properties, to the occurrence of different activities.

Three axioms where generated to evaluate the types of questions asked; the questioning technique comment, the missed follow-up question comment and the questioning efficiency
The questioning technique comment determines if the student has selected a question that matches one of the defined good or bad questioning styles. To do so the questions are assigned a component word property that matches one of these styles and the evaluation axioms uses this property to generate the comment. The missed follow-up question uses the answer type property to determine if a question provides a path to an SA element through its follow-up questions. If the student did not select a follow-up question that provided an SA element, the evaluation comment is generated. Finally the questioning efficiency comment compares how many questions the student has asked versus the number of SA elements they have obtained.

Two axioms were generated to evaluate how the student interacts with SA elements; the wrong SA element classification comment and the missed visual SA element comment. When a student obtains an SA element they are required to identify what type of trigger the element supports. If this proposed classification type does not match the correct classification type an evaluation comment is generated. As the student is investigating the visual scene they must click on visual SA elements to obtain them. The amount of time the students mouse spends within the defined boundary area is monitored and once this exceeds a threshold a comment is generated.

These axioms were generated in both first order logic and, to support the end goal of a real-time DSS, within Prolog. Following the development of the axioms the ontology was verified in both Prover9 and by developing a prototype DSS and user interface against a set of intended models. The evaluation found that the open-world nature of Prover9, and its difficulty with math formulas, were not conducive to the end goal of a real-time DSS. The Prolog implementation of the evaluation ontology proved much more successful and was verified in both a stand-alone Prolog implementation and within the inference engine of the DSS. Following the verification of the DSS behaviour a more complicated scenario was developed for the student user interface that evaluated the performance of the DSS under real life loads. Several improvements were made to the ontology implementation that significantly improved its speed.

13 Future Work

The evaluation ontology presented in this project was designed to extract IED specific information through the activity of questioning a witness. The axioms proposed in the ontology are an initial step and future work will be to validate the situations under which evaluation comments are generated and to generate more complex or situational specific comments. To
validate the generation of evaluation comments the DSS prototype should be presented to the instructors at the CF IED disposal operators course at Canadian Forces Base (CFB) Gagetown. Their initial feedback has been to request the addition of comments that deal with IED-D specific behaviour such as who should be questioned first. Additional development of evaluation comments can come from a comprehensive literature review into how questioning technique should be evaluated.

However, the concept of obtaining information through questioning is not unique to IED disposal and can be equally applied to other domains that deal with investigation such as doctors or police. The following section will describe how the evaluation ontology proposed in this paper could be modified for application to other domains.

13.1 Other Application Domains

The process of asking questions to obtain information is a generic concept that can be applied to many domains. To apply the evaluation ontology discussed in this project to other questioning domains, such as a doctor obtaining a patient history or police officer questioning a suspect or witness, it is necessary to separate the IED specific concepts from the generic questioning concepts that are consistent across domains and to develop a generic questioning evaluation ontology that could be extended into different domains.

In the IED specific ontology presented here deals with questioning witnesses to obtain SA elements that then need to be classified to help identify a device trigger type. The concept of SA elements and classification of a device trigger are clearly specific to the domain and can be ignored, however the concept of SA elements maps well to a generic piece of information or knowledge that needs to be obtained from a witness. The generic questioning ontology should introduce a more generic information concept in the taxonomy that sits between the PSL object and the IED specific SA element. Different domains could then introduce additional, domain specific concepts similar to the SA element. The current questioning efficiency evaluation axiom would then be modified to look at the number of information objects revealed as opposed to the number of SA elements revealed. The current SA element specific axiom would continue in the IED domain but would not be applicable to the other domains and hence would not be included in the generic questioning ontology.
The ontology concept of a question introduced in the IED evaluation ontology could be moved directly into the generic questioning evaluation ontology. The background material used to generate the evaluation axioms [4] deals with generic questioning technique and not with the IED specific domain, as such the comments are applicable in any questioning domain.
References


# Appendices

## Appendix A  Intended Models

### A-1 Intended Model Inputs

<table>
<thead>
<tr>
<th>Activity Occurrences</th>
<th>occurrence_of(qOcc1, questionActivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>occurrence_of(qOcc2, questionActivity)</td>
</tr>
<tr>
<td></td>
<td>occurrence_of(qOcc3, questionActivity)</td>
</tr>
<tr>
<td></td>
<td>occurrence_of(qOcc4, questionActivity)</td>
</tr>
<tr>
<td></td>
<td>occurrence_of(qOcc5, questionActivity)</td>
</tr>
<tr>
<td></td>
<td>occurrence_of(eOcc1, classifySAElementActivity)</td>
</tr>
<tr>
<td></td>
<td>occurrence_of(eOcc2, classifySAElementActivity)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question Properties</th>
<th>hasAnswerType(q1, Neutral)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hasAnswerType(q2, Partial)</td>
</tr>
<tr>
<td></td>
<td>hasAnswerType(q3, Neutral)</td>
</tr>
<tr>
<td></td>
<td>hasAnswerType(q4, Partial)</td>
</tr>
<tr>
<td></td>
<td>hasAnswerType(q5, Complete)</td>
</tr>
<tr>
<td></td>
<td>hasComponentWordProperty(q1,Biased)</td>
</tr>
<tr>
<td></td>
<td>hasComponentWordProperty(q2,None)</td>
</tr>
<tr>
<td></td>
<td>hasComponentWordProperty(q3,Jargon)</td>
</tr>
<tr>
<td></td>
<td>hasComponentWordProperty(q4,None)</td>
</tr>
<tr>
<td></td>
<td>hasComponentWordProperty(q5,VisualAids)</td>
</tr>
<tr>
<td></td>
<td>parent(q1,q1)</td>
</tr>
<tr>
<td></td>
<td>parent(q2,q1)</td>
</tr>
<tr>
<td></td>
<td>parent(q3,q1)</td>
</tr>
<tr>
<td></td>
<td>parent(q4,q1)</td>
</tr>
<tr>
<td></td>
<td>parent(q5,q4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SA Element Properties</th>
<th>correctClassification(e1, Timer)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>proposedClassification(e1, Command)</td>
</tr>
<tr>
<td></td>
<td>correctClassification(e2, Victim)</td>
</tr>
<tr>
<td></td>
<td>proposedClassification(e2,Victim)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Participation</th>
<th>participates_in(q1, qOcc1, t1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>participates_in(q2, qOcc2, t2)</td>
</tr>
<tr>
<td></td>
<td>participates_in(q3, qOcc3, t3)</td>
</tr>
<tr>
<td></td>
<td>participates_in(q4, qOcc4, t4)</td>
</tr>
<tr>
<td></td>
<td>participates_in(q5, qOcc5, t5)</td>
</tr>
<tr>
<td></td>
<td>participates_in(e1,eOcc1,t6)</td>
</tr>
<tr>
<td></td>
<td>participates_in(e2,eOcc2,t7)</td>
</tr>
</tbody>
</table>
### Time Ordering

| before(t1, t2) |
| before(t2, t3) |
| before(t3, t4) |
| before(t4, t5) |

### A-2 Intended Model Outputs

#### Activity Occurrences

- occurrence_of(cOcc1, commentActivity)
- occurrence_of(cOcc2, commentActivity)
- occurrence_of(cOcc3, commentActivity)
- occurrence_of(cOcc4, commentActivity)
- occurrence_of(cOcc5, commentActivity)

#### Comment Properties

- questioningTechniqueComment(c1)
- questioningTechniqueComment(c3)
- questioningTechniqueComment(c4)
- missedFollowupQuestionComment(c2)
- wrongSAElementClassificationComment(c5)

- hasComponentWordProperty(c1, Biased)
- hasComponentWordProperty(c3, Jargon)
- hasComponentWordProperty(c4, VisualAids)

#### Participation

- participates_in(c1, cOcc1, t1)
- participates_in(c2, cOcc2, t3)
- participates_in(c3, cOcc3, t3)
- participates_in(c4, cOcc4, t5)
- participates_in(c5, cOcc5, t6)
Appendix B  Prover9 Input File

%Supporting Axiom 1
%Instantaneous Occurrence
all x all y
    ( (askQuestion(x) | generateComment(x) | classifySAElement(x) ) & occurrence_of(y,x)
     -> ( beginof(y) = endof(y) )).

%Supporting Axiom 9
%Subclass Axiom
all x
    (comment(x) | question(x) | saElement(x) | property(x)
     ->object(x)).

%Supporting Axiom 6
%Subclass Axiom
all x
    (comment(x)
     -> -(question(x) | saElement(x) | property(x))).

%Supporting Axiom 6
%Sort Constraint Axiom
all x
    (question(x)
     -> -( property(x) | saElement(x) )).

%Supporting Axiom 6
%Sort Constraint Axiom
all x
    (property(x)
     -> -( saElement(x) )).

%Subclass Axiom
all x
    (questioningTechniqueComment(x) | missedFollowupQuestionComment(x) | wrongSAElementClassificationComment(x)
     ->comment(x)).

%Subclass Axiom
all x
(componentWord(x) |
  answerType(x) | classification(x) |
  ->property(x)).

%Supporting Axiom 7
%Sort Constraint Axiom
all x
  (componentWord(x) |
   ->
   -( answerType(x) | classification(x) )).

%Supporting Axiom 7
%Sort Constraint Axiom
all x
  (answerType(x) |
   ->
   -( classification(x) )).

%Supporting Axiom 8
%Sort Constraint Axiom
all x
  (questioningTechniqueComment(x) |
   ->
   -( missedFollowupQuestionComment(x) | wrongSAElementClassificationComment(x) )).

%Supporting Axiom 8
%Sort Constraint Axiom
all x
  (missedFollowupQuestionComment(x) |
   ->
   -( wrongSAElementClassificationComment(x) )).

%Supporting Axiom 10
%Subclass Axiom
all x
  (askQuestion(x) | generateComment(x) | classifySAElement(x) |
   ->activity(x)).

%Sort Constraint Axiom
all x
  (askQuestion(x) |
   ->
   -(generateComment(x) | classifySAElement(x))).

%Sort Constraint Axiom
all x
  (generateComment(x) |
classifySAElement(x).

%Supporting Axiom 5
%Sort Constraint Axiom
all x all y
  (parent(x,y)
   -> question(x) & question(y)).

%Sort Constraint Axiom
all x all y
  (hasComponentWordProperty(x,y)
   ->
    ( (question(x) | questioningTechniqueComment(x))
      & componentWord(y) )).

%Sort Constraint Axiom
all x all y
  (hasProposedClassification(x,y)
   ->
    ( (saElement(x) & classification(y) )).

%Sort Constraint Axiom
all x all y all x
  (hasCorrectClassification(x,y)
   ->
    ( (saElement(x) & classification(y) )).

%Supporting Axiom 3
%Existence Axiom
all x
  (question(x) | questioningTechniqueComment(x)
   ->
    ( exists y
      (hasComponentWordProperty(x,y) & componentWord(y)) )).

%Supporting Axiom 4
%Unique Existence Axiom
all x all y all x
  ( hasComponentWordProperty(x,y) &
    hasComponentWordProperty(x,z)
   ->
    ( y = z )).

%Unique Existence Axiom
all x all y all x
  ( hasProposedClassification(x,y) &
    hasProposedClassification(x,z)
%Unique Existence Axiom
all x all y all x
(hasCorrectClassification(x,y) &
hasCorrectClassification(x,z)
->
(y = z)).

%Supporting Axiom 3
%Existence Axiom
all x
(question(x)
->
(exists y
(hasAnswerType(x,y) & answerType(y)))).

%Supporting Axiom 3
%Unique Existence Axiom
all x all y all x
(hasAnswerType(x,y) & hasAnswerType(x,z)
->
(y = z)).

%Sort Constraint Axiom
all x all y
(hasAnswerType(x,y)
->
(question(x) & answerType(y))).

%Supporting Axiom 2
%Follows Axiom
all x all y
(before(beginof(x),beginof(y)) &
activity_occurrence(x) &
activity_occurrence(y) &
(! (exists z
(activity_occurrence(z) &
between(beginof(x),beginof(z),beginof(y)))))
<-> follows(y,x)).

%Questioning Technique Axiom
all p all q all t all a1 all o1
((question(q) & occurrence_of(o1,a1) &
askQuestion(a1) & participates_in(q,o1,t) &
hasComponentWordProperty(q,p))
-> (exists c exists o2 exists a2
( questioningTechniqueComment(c) & occurrence_of(o2,a2) &
  generateComment(a2) & hasComponentWordProperty(c,p) &
  participates_in(c,o2,t) )).

all c all o2 all a2 all p
  ( (questioningTechniqueComment(c) & occurrence_of(o2,a2) &
  generateComment(a2) & hasComponentWordProperty(c,p) &
  participates_in(c,o2,t))
  -> ( exists q exists o1 exists a1
  (question(q) & occurrence_of(o1,a1) &
   askQuestion(a1) & participates_in(q,o1,t) &
   hasComponentWordProperty(q,p) ))).

%Missed Followup Question
all q1 all t1 all a1 all o1 all q2 all o2 all t2
  ( (question(q1) & question(q2) & occurrence_of(o1,a1) &
  occurrence_of(o2,a1) & askQuestion(a1) &
  participates_in(a1,o2,t2) & participates_in(a1,o1,t1) &
  follows(o2,o1) & -parent(q1,q2) &
  hasAnswerType(q1,Partial))
  -> ( exists c exists o3 exists a2
  ( missedFollowupQuestionComment(c) &
   generateComment(a2) &
   occurrence_of(o3,a2) &
   participates_in(c,o3,t2) ))).

all c all o3 all a2 all t2
  ( (missedFollowupQuestionComment(c) &
  generateComment(a2) &
  occurrence_of(o3,a2) &
  participates_in(c,o3,t2))
  -> ( exists q1 exists q2 exists a1 exists o1
  exists o2 exists t1
  ( question(q1) & question(q2) &
   occurrence_of(o1,a1) & occurrence_of(o2,a1) &
   askQuestion(a1) & participates_in(a1,o2,t2) &
   participates_in(a1,o1,t1) &
   follows(o2,o1) & -parent(q1,q2) &
   hasAnswerType(q1,Partial) ))).

%Wrong SA Element Classification
all c all e all t all a1 all o1
  ( (saElement(e) & occurrence_of(o1,a1) &
  classifySAElement(a1) & participates_in(e,o1,t) &
  hasProposedClassification(c,e) & -
  hasCorrectClassification(c,e) )

  -> ( exists q1 exists q2 exists a1 exists o1
  exists o2 exists t1
  ( question(q1) & question(q2) &
   occurrence_of(o1,a1) & occurrence_of(o2,a1) &
   askQuestion(a1) & participates_in(a1,o2,t2) &
   participates_in(a1,o1,t1) &
   follows(o2,o1) & -parent(q1,q2) &
   hasAnswerType(q1,Partial) ))).
\[
\rightarrow (\exists c \exists o_2 \exists a_2
\quad (\text{wrongSAElementClassification}(c) \&
\quad \text{occurrence}_\text{o}(o_2,a_2) \&
\quad \text{generateComment}(a_2) \& \text{participates}_\text{i}(c,o_2,t)))).
\]

\[
\forall c \forall o_2 \forall a_2
\quad (\text{wrongSAElementClassification}(c) \& \text{occurrence}_\text{o}(o_2,a_2) \& \text{generateComment}(a_2) \& \text{participates}_\text{i}(c,o_2,t))
\rightarrow (\exists e \exists o_1 \exists a_1 \exists c
\quad (\text{saElement}(e) \& \text{occurrence}_\text{o}(o_1,a_1) \&
\quad \text{classifySAElement}(a_1) \& \text{participates}_\text{i}(q,o_1,t) \&
\quad \text{hasProposedClassification}(e,c) \& \neg
\quad \text{hasCorrectClassification}(e,c))).
\]
Copyright Acknowledgements

This report contains IP material generated for DRDC-Toronto under contract number W7711-098153/002/TOR. All work and associated IP within this report remains under the copyright of Her Majesty the Queen in Right of Canada, as represented by the Minister of Defence, 2011.