A Scenario-Directed Computational Framework To Aid Decision-Making And Systems Development

A Thesis
Presented to
The Academic Faculty

by

Reginald L. Hobbs

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
August 2005

A Scenario-Directed Computational Framework To Aid Decision-Making And Systems Development

Approved by:

Date Approved: August 2005

To Lady MacBeth,

thank you for standing by me in this long,

but fruitful journey.

PREFACE

Back Off Man, I'm A Scientist! - Dr. Peter Venkman (Bill Murray), Ghostbusters

ACKNOWLEDGEMENTS

No one who achieves success does so without acknowledging the help of others.

The wise and confident acknowledge this help with gratitude. – Unknown

As I celebrate the silver anniversary of my association with the Georgia Institute of Technology, it is fitting that I reach this milestone by achieving the pinnacle that this fine instititution has to offer. Through 25 years of growth, I have gone from undergraduate student to airman in the US Air Force to systems administrator to adjunct professor. Through all these varied experiences, I have remained a Tech Man.

During my time as both a doctoral student and instructor, I've been approached by my students asking why one seeks a graduate degree. In keeping with my research on the use of narrative, my explanation consisted of a story on the motivation for learning. The story involves three characters, all seekers of truth. The first actor is an undergraduate who approaches the well of knowledge with a large cup, hoping to fill it to the brim with the stuff of learning. The contents enable the bearer to be a useful practitioner in his or her field, capable of dispensing knowledge whenever it is needed. The second actor, the master's graduate, encounters the well using a large bucket that also acts as a filter. This bucket contains not only a much larger amount of information, but concentrated from a specific area of the well. The master's graduate is truly an expert in their chosen discipline. The final protagonist is the doctoral candidate. He is motivated by the desire to create cups, design buckets, and build (and fill) wells.

My well building journey has been subsidized by many people who have placed my feet on the path of discovery. First and foremost, I thank God for giving me the opportunity to seek my doctorate and the ability to achieve it. Each challenge has only strengthened my desire to excel and made each surmounted obstacle more satisfying. I am humbled by His grace.

I would like to thank Dr. Colin Potts for his advice and for nurturing the spark of a researcher that existed under a pile of wet kindling. Our many discussions enabled me to think critically, ask the right questions, and seek the right answers. In keeping with the european model for doctoral candidacy, I truly consider myself the world's expert in my particular niche of computer science.

To Dr. Jim Gantt, for recognizing my potential and convincing me to become a scientist. Your support, both emotionally and professionally, has been immeasurable. You have been a mentor, in every sense of the word.

To my AIRMICS colleagues: Mr. Glenn Racine, Dr. Gerard McCord, Dr. Michael Evans, Mr. Butch Higley, Dr. Mark Kindl, Dr. John (Jay) Gowens, Dr. Adrienne Raglin, Mr. Binh Nguyen, Mr. Son Nguyen, Mr. Brian Rivera and others, thanks for showing what a good and successful research organization feels like and that the phrase "government worker" is not an oxymoron. To my administrative support, Mrs. Patsy Riley, Mrs. Delois Rogers, and Ms. Kim Brinkley, thank you for your ability to cut through the red tape and leap the government beauracracy in a single bound.

To Colonel Anthony Love, Colonel Mark Kindl (Ret.), Major Jeffrey (Jay) Reddick, Lieutenant Colonel Rachel Borhaurer, Colonel Ron Byrnes, Major David Dampier (Ret.) and the other military officers that I've had occasion to work with side by side, thanks for showing that the Army can develop critical thinkers and learn how to use them effectively. You've shown me that my research can have a positive impact on both the scientific community and my country.

To my supervisors at the Army Research Laboratory, particular Mrs. Pat Jones and Dr. Rick Helfman, thanks for your patience in realizing the potential ROI (Return On Investment) that rewards ARL. My carbon-to-diamond transformation into a researcher may well have benefitted from the additional pressure and heat supplied by ARL.

To my doctoral committee: Dr. Leo Mark, Dr. Mary Jean Harrold, Dr. Spencer Rugaber, and Dr. David Dampier. Thank you for your insight and assistance in manipulating my research ideas into a more palatable form for the larger research community. I'd like to thank Dr. Mark, in his capacity as PhD program coordinator, for helping with what could have been a very awkward transition.

To my advisor, Dr. Melody Moore, I extend my special gratitude for her advocacy and support. It is a major challenge to establish a mentorship in the later stages of a doctoral program, but you rose to that challenge admirably. Results are the best indicator of your impact; 4 months after you became my advisor, I was doing my defense of research. I was motivated as much by your example and work ethic as by your positive view of my work. I look forward to merging research ideas and having a collaborative impact on the research community.

To my mother and father, thanks for your belief in what I could eventually accomplish. Mom, I definitely inherited my fortitude and desire to attempt new things from you. I also inherited a stubbornness that made it impossible for me to quit something I started. Pop, thanks for being a good example of fatherhood and how to keep a family going no matter what happens.

To my brother, Rodney, and my sisters, Belinda and Tonja, thanks for being proud of me. It does make a difference to me what you think. To my brother, your accomplishment as a soldier has made me want to have an even greater impact for my country. If my research ends up helping the army, it is because of my desire to have it help you.

To my children: Jason, Shikitha, Justin, Jenna, Gino, Jordi, and Jules, thank you for being an even greater accomplishment for me than anything else that I've ever done. I especially want to let Julian, who has never known what it is like for daddy not to be a graduate student, know that I'm now going to be there for you even more. For all my children, thank you for being patient while I sought this doctorate; you've been a large part of my motivation as well.

To my loving wife Lisa. It is a bit cliched and not sufficiently strong just to quote that "behind every successful man, there is a strong woman". You've been more than someone behind me pushing; you've been in front of me pulling, beside me nudging, and all around me cheering me on. It is a cosmically huge understatement to say that I couldn't have done this without you. I love you.

Contents

List of Tables

List of Figures

LIST OF SYMBOLS OR ABBREVIATIONS

ACL	Agent Communication Language.
ACM	Association for Computing Machinery.
ACML	Agent Communication Markup Language.
AIRMICS	Army Institute for Research in Management Information, Communications, and Computer Science.
ARL	Army Research Laboratory.
ASCII	American Standard Code for Information Interchange.
BP	Business Process.
BRML	Business Rules Markup Language.
C4I	Command,Control,Communications,Computers, and Intelligence.
CACM	Communications of the ACM.
CALS	Continuous Acquisition and Life-Cycle Support.
CBML	Case-Based Markup Language.
CBR	Case-Based Reasoning.
CFG	Context Free Grammar.
CFP	Call For Participants.
CGI	Common Gateway Interface.
CHI	Computer-Human Interaction.
CLP	Courteous Logic Program.
COE	Common Operating Environment.
CSCW	Computer-Supported Cooperative Work.
CSS	Cascading Style Sheet.
CTA	Cooperative Technical Agreement.
DAML	DARPA Agent Markup Language.
DAML-ONT	DAML Ontology.
DARPA	Defense Advanced Research Projects Agency.
DBI	Database Interface.
DIF	Document Interchange Format.

DISA	Defense Information Systems Agency.
DL	Descriptive Logic.
DoD	Department Of Defense.
DOM	Document Object Model.
DPML	Design Pattern Markup Language.
DTD	Document Type Definition.
DV	Dependent Variable.
ERD	Entity Relationship Diagram.
EU	Expected Utility.
FIPA	Foundation For Intelligent Physical Agents.
GML	Generalized Markup Language.
GUI	Graphical User Interface.
HCI	Human-Computer Interaction.
HLA	High-Level Architecture.
HTML	Hypertext Markup Language.
HVAC	Heating, Ventilation, and Air-Conditioning.
ICSE	International Conference on Software Engineering.
IEEE	Institute of Electronics and Electrical Engineering.
IRB	Institute Review Board.
IS	Information System.
ISO	International Standards Organization.
IT	Information Technology.
IV	Independent Variable.
JDBC	Java DataBase Connectivity.
JTA	Joint Technical Architecture.
KIF	Knowledge Interchange Format.
KQML	Knowledge Query Markup Language.
KR	Knowledge Representation.
KSE	Knowledge Sharing Effort.

LALR	Look Ahead Left to Right.
LC	Line Of Contact.
LD	Line Of Departure.
LIFO	Last-In, First-Out.
MDA	Model-Driven Architecture.
MIT	Massachusetts Institute of Technology.
ModSAF	Modular Semi-Autonomous Forces.
MOTW	Missions Other Than War.
M&S	Modeling & Simulation.
MSU	Mississippi State University.
MUD	Multi-User Dimension/Dungeon.
NIST	National Institute for Standards and Technology.
ODU	Old Dominion University.
OIL	Ontology Inference Layer.
OMT	Object Model Template.
OneSAF	One Semi-Autonomous Forces.
OWL	Ontology Web Language.
PERL	Practical Extraction and Reporting Language.
RDF	Resource Description Framework.
RDFS	Resource Description Framework Schema.
RPG	Role Playing Game.
ScenIC	Scenario Inquiry Cycle.
SCESM	Scenarios and State Machines.
SCML	Scenario Markup Language.
SE	Software Engineering.
SGML	Standard Generalized Markup Language.
SIGCHI	Special Interest Group on Computer-Human Interaction.
SIMCI OIPT	Simulations-to-C4I Overarching Integrated Product Team.
SISC	Simulations Interoperability and Standards Committee.

SISO	Simulations Interoperability and Standards Organization.
SIW	Simulations Interoperability Workshop.
SME	Subject Matter Expert.
SOAP	Simple Object Access Protocol.
SOM	Simulation Object Model.
SOP	Standard Operating Procedure.
SQL	Standard Query Language.
TTP	Tactics, Techniques, and Procedures.
UML	Unified Modeling Language.
UTL	Universal Task List.
XLL	XML Linking Language.
XMI	XML Metadata Interchange.
XML	Extensible Markup Language.
XMLS	XML Schema.
XMSF	Extensible Modeling and Simulation Framework.
XSL	XML Style Language.
XSLT	XML Style Language Transformation.
YACC	Yet Another Compiler Compiler.

GLOSSARY

acteme	The lowest level of *hypertext* activity, such as following a link, p. 19.
action	1)A change in environment. 2) The activity being filmed, p. 13.
actor	1) A character that initiates action. 2) A participant, p. 16.
archetype	1) A character chosen by the storyteller to have a specialized, non-standard characteristic in addition to the norm. (e.g. a reluctant hero) 2) Deep and abiding patterns in the human psyche that remain powerful and present over time, p. 15.
audience	The body of listeners or spectators that experience a story, p. 12.
Character	A bundle of characteristics, behaviors, and motivations, that when taken together define a collection of entities with those characteristics (e.g. hero, villain, damsel-in-distress), p. 34.
conclusion	An exit point from an episode based on actions that establish post-conditions for the episode, p. 47.
continuity	1) Maintained consistency, either temporal or physical, within a story. 2) An uninterrupted succession or unbroken course, p. 13.
decision	A determination arrived at after consideration; a conclusion, p. 1.
decision-making	The process of determining from among alternatives, p. 1.
dependency	An association representing movement from an action to the next logical action, p. 45.
engagement	The state of mind attained by the audience in order to experience a story, p. 12.
equivalence	An association among two actions within a set of equivalent actions that have the same meaning within a scenario. (e.g. *pick up* the ball, *lift* the ball), p. 45.
event	1) An ordered collection of actions representing something significant. 2) An occurrence, especially a significant one, p. 11.
flashback	1) A technique of showing past happenings in order to expand an audience's comprehension of the present story, character, or situation. 2) Shots or sequences that convey events or information, which precede the time established as the present in a film, p. 46.
genre	A classification of story using the tradition within which the story is set as it relates to previously made stories(for example, horror, science fiction, drama), p. 16.
goal	1) An objective to successfully meet a specific condition. 2) Outer motivation, p. 11.

hyperscenario A scenario document containing link structures for navigation to other documents, p. 4.

hypertext Representations of a story using a document in which interactive structural operations are intermingled with text, p. 66.

mechanics The representation of a story chosen by the storyteller (e.g. prose, storyboard, outline), p. 13.

mise en abyme Any aspect enclosed within a work that shows a similarity with the work that contains it, p. 17.

mise en scene The manipulation of all the elements that contribute to the filming of a scene; staging, p. 18.

morphology The study of forms, p. 12.

motivation 1) The purpose for the actions of a character. 2) Character in action, p. 15.

multiform A narrative that presents a single situation or plotline in multiple versions, versions that would normally be mutually exclusive, p. 49.

narrative 1) A narrated account; a story. 2) Represents a specific navigation through a universe of story elements, p. xxiv.

ontology A formal explicit description of concepts in the domain of discourse, the properties of each of those concepts and attributes, and restrictions on those properties, p. 27.

perception Understanding or insight into the story by the audience, p. 12.

prerequisite An association denoting a dependency between the current episode and the previous episode, p. 46.

Prop A character that is the target of an action, p. 35.

protagonist The character that drives the plot and initiates the action, p. 15.

rewind A reversely ordered collection of previous actions within a shot from the current action in a scene, p. 46.

scenario 1) A form of narrative as a collection of episodes with the purpose of describing the actions of characters within an environment. 2) A proposed specific use of a system; a description of one or more end-to-end transactions involving the required system and it's environment. 3) A single path through a use case, one that shows a particular combination of conditions within that use case, p. 2.

Scene A part of an episode that occurs within a particular setting, p. 44.

scope The extent of an activity, situation, or subject, p. 13.

screenplay A motion picture script, p. 16.

script	1) A document that describes an appropriate story sequence in a particular context. 2) A structure that describes an appropriate sequence of events in a particular context, p. 16.
sequence	An ordered collection of episodes representing a meaningful segment of the scenario, p. 47.
Setting	Fixes a story in time and space, p. 16.
setup	An entry point into an episode based on actions that establish preconditions for the episode, p. 47.
slide	An association between an action within a scenario to an alternate scenario, p. 47.
story	A composition of events, instances of characters, and entities and the interactions between them, p. 12.
storyteller	The presenter of a story to an audience, p. 12.
style	The method of telling the story chosen by the storyteller, p. 13.
theme	A universal statement about the human condition the storyteller is making; the ideas and words that the storyteller wishes the audience to walk away with, e.g. "good triumphs over evil"), p. 17.
Transition	An association between a scene and the next logical scene within an episode, p. 47.
use case	A typical interaction between a user and a computer system, p. 8.

SUMMARY

Scenarios are narratives that illustrate future possibilities or existing systems, and help policy makers and system designers choose among alternative courses of action. Scenario-based decision-making crosses many domains and multiple perspectives. Domain-specific techniques for encoding, simulating, and manipulating scenarios exist, however there is no general-purpose scenario representation capable of supporting the wide spectrum of formality from executable simulation programs to free-form text to streaming media descriptions. The claim of this research is that there is a computer readable scenario framework that can capture the semantics of a problem domain and make scenarios an active part of decision making. The challenge is to define a representation for scenarios that supports a wide range of discussion and comprehension activities while remaining independent of content and access mechanisms. This dissertation describes a scenario ontology derived by examining alternate forms of narrative: thought experiments, mental models, case-based reasoning, use cases, design patterns, screenwriting, film-editing, intelligent agents, and other narrative domains. The scenario conceptual model was based on the an analysis of forms of narrative and the activities of storytelling. This method separates what a narrative is from how it is used. The research contribution is the development of the hyperscenario framework. A hyperscenario is a scenario representation containing link structures for navigation between scenario elements. The hyperscenario framework consists of the scenario ontology, scenario grammar, and a scenario specification called Scenario Markup Language (SCML). The results of the web-enabled simulation experiment validate the improvement on decision-making due to the hyperscenario framework.

Chapter 1

INTRODUCTION

Creativity, as has been said, consists largely of rearranging what we know in order to find out what we do not know. Hence, to think creatively, we must be able to look afresh at what we normally take for granted.

– *George Keller*

1.1 Problem Description

The problem domain addressed by this research is that of scenario-based design and decision-making. A *decision* is a determination arrived at after consideration; a conclusion [126]. A decision is related to the concept of selecting a choice or alternate. *Decision-making* is the process of determining from among alternatives. There have been many studies establishing the importance and validity of scenario analysis for systems development [25]. Requirements engineering, prototyping, high-level design, even maintenance activities benefit greatly from "what-if" studies and testing based on deriving scenarios [119]. There are different diagramming techniques for depicting scenarios, such as sequence/collaboration diagrams of use cases [57]. The problem with such notations and formalized descriptions is that they are passive parts of the design process. They are very useful as points of discussion for generating ideas and prototyping, but there is not necessarily any direct connection with other design artifacts.

This disconnect makes it difficult to maintain the relationship between design decisions and the scenarios that generated them. System developers need to capture design decisions and the semantics of the problem domain in computer readable form. Formalized models may force design activities to be captured using a terminology unfamiliar to the problem domain or unnecessarily restricted. Finally, studies on thought experiments and mental models have shown that problem-solving activities benefit greatly from the ability to manipulate and reform decision elements [99].

The purpose of this research is to develop a computational representation of narrative to aid policy makers, systems developers, and trainers who need to ask such "what-if" questions in their decision making. This involves (1) a theoretical investigation of the nature of narrative in decision making and (2) an implementation of a computational framework for narrative.

1.2 Motivation

Consider the following situations:

- A visitor from out-of-town stops to ask directions from Hartsfield Atlanta Airport to the campus at Georgia Tech. A knowledgeable resident explains how to get there by describing the navigation from the point of view of the visitor: First you get onto the interstate going North, then take exit 100.
- A software developer creates a prototype user interface for a word processing package and discusses it with a potential user by asking what type of tasks are performed.
- An army commander has a tactical planning briefing with his staff where they discuss alternate ways of meeting mission objectives by creating a storyboard of tactics.
- A stock market simulation is a mathematical model. Market fluctuations and the factors that cause them are predicted by the outcome of the simulation.

Each of these situations is an example of a scenario. A *scenario* is an instance of a system behavior, a sequence of events and interactions that describe a specific case of a system's function [87]. Whether it is for the purpose of comprehension (out-of-town visitor),design (user interface prototype), collaborative decision making (Army tactical planning briefing), or dependency analysis (stock market simulation), a scenario is a powerful method for discussing system behavior.

Many practical disciplines make use of scenarios to improve decisions. System designers and policy makers use scenarios to assist them in making design and policy decisions [26].

Scenarios are used throughout software development to examine alternate design decisions and requirements [111]. *Requirements engineering*, the process of determining requirements for a proposed system, has used reasoning techniques about scenarios to help generate and evaluate specifications [145]. Throughout the 1990's, the design research communities in human-computer interaction, object-oriented software design, and business administration have paid concerted attention to user-centered design representations that utilize scenarios [135]. Scenarios are important in management and enterprise modeling. Scenario-based risk assessment is an effective cost-benefits analysis technique for determining future directions of an organization. A major technique in business process re-engineering is the creation of scenarios that illustrate how a business actually works and how its operations might be improved [140]. Some of these disciplines use methods in which established categories of scenarios are design artifacts in their own right (for example, storyboards as designs for user interfaces, sequence diagrams as specifications of software behavior, business scenarios as planning documents, etc.).

Not only are scenarios of practical importance, they appear to play a fundamental role in comprehension. Cognitive scientists, and social scientists who study everyday and professional reasoning (e.g. by scientists) claim that narrative is central to processes of explanation, inference, and interpretation [137], and apply this theory to professional practices such as pedagogy and educational technology. Scenarios codify and externalize algorithmic mental models and thought experiments [100]. The decision-maker may test hypotheses by constructing what-if scenarios on top of a baseline description of the situation under consideration. Whether these scenarios are merely described, enacted through role-playing, or simulated computationally, whether they faithfully represent a real phenomenon or merely provide the outline sketch of a possible course of action, the intention is the same: to clarify the relationships among actions and features of a situation and to understand better the consequences of actions.

1.3 Research Questions

The primary research question addressed in this work is: “What is a scenario, with respect to narrative and decision-making?” In other words, what is the scenario ontology necessary to describe the essential building blocks for decision-oriented storytelling? This scenario ontology is not derived as a union of all possible story elements across the aforementioned domains, but instead, as an intersection of those common primitives that appear to cross boundaries.

The second research question concerns the form of the scenario specification. How can scenarios be represented in a computer readable form that is non-intrusive, semi-formal, extensible, and portable, representing the semantics and modalities of storytelling? This computer-readable form comes about by treating scenario artifacts as documents. These scenario documents can be authored, shared, and analyzed using the appropriate software tools. Creating the documents as hyperscenarios makes it possible to create different representations of the same scenario content. A *hyperscenario* is a scenario document containing link structures for navigation to other documents. These links among text, graphics, and other multimedia elements supports multiple perspectives, representations, and collaborative development of artifacts.

The final research question is: Does the availability of the computer-readable scenario framework improve decision-making? Given a story form of a problem domain during a decision-making task, is there a noticeable improvement in the quality and the effectiveness of decision-making?

The first two research questions are answered using direct research methods, using an analysis of narrative domains and the development of the scenario ontology, respectively. The final question is answered empirically, using a simulation experiment to assess the improvement on decision-making.

1.4 Research Contribution

The primary contribution of this research is the Hyperscenario Framework. The *Hyperscenario Framework* consists of the scenario ontology, scenario grammar, and a scenario language called Scenario Markup Language (SCML). An extensive literature review of narrative forms identified the concepts and classes that define scenarios. An explicit mathematical model describing scenario elements and operations is defined using set theory and logic constructs. The context-free grammar derived using the rules in the scenario conceptual model was implemented into SCML.

1.5 Thesis Statement

The claim of this research is that the Hyperscenario Framework captures the semantics of a problem domain and improves decision making.

1.6 Overview of Remaining Chapters

Chapter 2 of this dissertation covers background information and related work. Chapter 3 is a detailed discussion of the scenario ontology including the conceptual model, formal mathematical description of scenario elements, grammar definition, and language implementation. Chapter 4 covers potential scenario applications that can use the features and characteristics of SCML. A prototype scenario generation tool consisting of a web-enabled game simulation is described in this chapter as well. Chapter 5 describes the design of the empirical study to validate the impact of a scenario framework on decision-making. The independent variable, dependent variables, metrics gathered, method of analysis, and techniques for mitigating threats to validity are outlined in the design chapter. Chapter 6 is an analysis of the results of the study, displaying and interpreting the data on decision-making for each experimental group. Finally, Chapter 7 concludes with a discussion of the future research on the Hyperscenario Framework.

Chapter 2

RELATED WORK

The structure of a play is always the story of how the birds came home to roost.

– Arthur Miller

This chapter is a literature review that outlines the narrative background of the Hyperscenario Framework. *Narrative* is: the representation in art of an event or story, also an example of such a representation [126]. Scenarios are forms of narrative concerned with problem-solving. Examining different forms of narrative and existing scenario applications helped to identify the elements that make up a scenario.

2.1 Narrative Applications and Research Efforts

The first section of this chapter describes tools and languages in business process analysis, case-based reasoning, and use cases. Each application used some form of narrative to help in the decision-making process.

2.1.1 Knowledge Representation for Business Rules

Intelligent agents are programs designed to retrieve and disseminate knowledge in a distributed fashion. The FIPA *(Foundation for Intelligent Physical Agents)* is a governing body that created standards for an *Agent Communication Language (ACL)* [56]. The standards outline principles and detailed requirements for agent communication, independent of their implementations. Grosof and Labrou, of IBM T. J. Watson research center, developed an *Extensible Markup Language (XML)* approach for intelligent agent communication [72]. Their research focused on two separate language implementations to support an agent-based environment: ACML and BRML.

ACML (Agent Communication Markup Language) is an XML encoding of the ACL standard. The semantics of FIPA ACL uses declarative knowledge representation format. *Declarative knowledge* representation uses speech-querying primitives, such as speaker, hearer, and communicative act. ACL message syntax uses additional parameters to represent high-level constructs. There are several pragmatic elements of the ACL syntax, such as parsing and querying, to describe the operational aspects of the agent messages. The large number of pragmatic conventions to adhere to minimizes the amount of interoperability between different systems that use the same ACL. ACML was an attempt to address these interoperability issues in an extensible form. Pragmatic/operational elements are more easily incorporated into the ACL syntax by using XML linking structures.

The second research effort encoded business rules content in a language called *Business Rules Markup Language (BRML)*. An e-commerce application that uses agents for bidding negotiations was chosen to illustrate the usefulness of BRML. The business rule representation for agent information was initially proposed by the *Knowledge Sharing Effort (KSE)*, a project funded by the *Defense Advanced Research Projects Agency (DARPA)* in the early 1990s. The goal of the KSE was to develop techniques, tools, and methods for knowledge sharing/reuse between knowledge based systems. In the KSE model, agents are treated as knowledge bases that exchange propositional information. The KSE model involved three layers corresponding to: 1) specifying the propositional attitudes, 2) specifying the proposition (i.e. the knowledge), and 3) the ontology (vocabulary). The KSE implementation of this model uses the *Knowledge Query and Manipulation Language (KQML)* [122] for propositional attitudes, *Knowledge Interchange Format (KIF)* [62] to represent propositions, and *Ontolingua* [74] for describing ontologies. KIF essentially displays first order logic. The BRML specification uses KIF propositions to represent business processes [72]. KIF is a content language that can show a broad range of rules, however, it suffers from two problems: it is logically monotonic and it is a pure belief knowledge representation. *Logical monotonicity* means that default rules and conflict handling cannot be expressed, making exceptions and priority rules hard to depict. *Pure belief* means the actions are invoked only as the result of rule inferencing, disallowing the use of external procedure calls for rule negotiation. BRML focused on correcting the monotonicity problem with *Courteous Logic*

Program (CLP) techniques for conflict resolution and prioritization. CLP's are declarative statements that depict a rule base. BRML is essentially an XML encoding of CLP rule sets.

2.1.2 Use Case Scenarios for Business Processes

The research project from the University of Paris focused on creating use case scenarios to describe business processes [140]. A *use case* is a unit of interaction between a user and a system; it is a meaningful unit of work [57]. A *Business Process (BP)* is a set of activities that produce an output valuable to the customer. BPs reside on different levels, based on requirements. The *organizational interactions* BP level is interactions between agents. The *system interactions* BP level describes how the *Information System (IS)* supports organizational activities. The final level, concerned with internal operations of the organization, is called the *System Internal* BP level. Because of the complexity of BPs and the potential number of agents that interact, the researchers chose an incremental design technique. They defined a set of rules for describing BP fragments, integrating the fragments into a single use case specification inclusive of all levels of BPs. The development effort was centered around authoring text documents with these use case scenarios as templates. The researchers used simple natural language processing techniques to capture use case semantics from text documents.

Conceptual information had to be gathered that could situate the business process within the organizational environment. This context was made up of the name of the BP, the initiating agent (entity that starts the business process) and the agents' goal for the BP. There is a set of preconditions defined as the initial state of a use case, with respect to a particular resource. There were two types of scenarios: normal and exceptional. The *normal* scenario describes a singular path of actions done to reach a final state that sufficiently fulfills the goal. An *exceptional* scenario is also a course of action, but the final state does not completely fulfill the goal of the initiating agent.

The relevancy of this use case project to the hyperscenario framework was the derivation of scenario fragments, in this instance, business processes. The researchers looked more at establishing rules for how these fragments could be captured from existing documentation instead of determining an appropriate representation. The natural language processing techniques and the terminology of the particular business organization being studied drove the specification of the use cases. Their approach limits the application of the technique to limited domain, i.e., in a business that uses the exact same description of its business processes. However, the authors did consider levels of details required within a use case. These levels begin to categorize BP use cases in terms of multiple views and perspectives, from a system and organizational standpoint. Knowing what difference it makes and how to transition levels treats use cases as dynamic artifacts of organizational knowledge.

2.1.3 Case Based Reasoning

Hayes and Cunningham, designers of the *Case Based Markup Language (CBML)*, focused on the development of an XML language for *Case-Based Reasoning (CBR)* [76]. CBR is a technique for problem solving which looks for previous examples that are similar to the current problem, particularly when heuristic knowledge is not available [108]. CBML was developed to support E-commerce applications. The CBR approach to business knowledge creates a reusable knowledge base. CBR data is integrated with the *Information Technology (IT)* infrastructure of a business using industry-specific vocabulary. For example, cases could be descriptions of the commodities on sale, such as package holidays, hardware configurations, and real estate. Hayes and Cunningham proposed CBML to support the distributed computing of cases. The initial form of CBML was an XML language that used a *Document Type Definition (DTD)* to represent the grammar. The researchers examined two existing CBR applications that used XML query relational databases. The first system, called the *Caret* system, was developer Hideo Shimazu in 1998 to retrieve cases from a database system [168]. The use of XML as a basis for the query format, along with a retrieval algorithm focused on the coded tags, minimized the number of records required to build case data. The second example of an XML representation of case data was used for a *Heating Ventilation and Air Conditioning (HVAC)* sales support system to query a Microsoft Access© database [185].

One problem with these approaches is the limited mechanism for data typing and feature weighting within DTDs. DTDs are also inflexible with respect to merging legacy representations. Version 1.0 of CBML was as designed using a DTD to represent the grammar. It was intended as a standard for exchanging information between classification and diagnostic systems that use CBR information. CBML required two files for each implementation: structure file and a base file. The purpose of the case *structure* file was to define the elements available to a developer within a specific domain - the features, types, weights, etc. that were appropriate. The case *base* file was used to describe elements available to an application instance. The weakness of the CBML architecture was that both documents were required for an application, although their relationship with one another was almost orthogonal. This is the reason that Hayes and Cunningham chose to modify the architecture to use XML namespaces and schemas to create a view of CBR data, as opposed to a specific grammar. *Namespaces* are a markup language identification technique that allows designers to uniquely qualify element names and relationships between them. Namespaces allow merging of alternate XML representations without element naming conflicts.

A difference between the CBML work and the Hyperscenario Framework is that their description of CBR information is not as expressive a form of narrative as a hyperscenario. The CBR information lists features and attributes associated with canned business transactions. Higher-level knowledge of case data, such as goals for the cases and multiple perspectives, were not being stored and retrieved. Also, there were very limited applications for the CBML representation, specifically information storage and retrieval and data exchange. Other ways of manipulating and authoring case data were not being explored.

The assessment of a DTD-based content representation in the CBML project was significant for the hyperscenario framework. The specification language of the Hyperscenario Framework is DTD grammar. An important issue for applications using the hyperscenario approach is the choice of mapping technique for a particular narrative domain. This mapping might necessitate an external file for domain information, similar to the structure/base file architecture initially used for CBML. Their lessons learned give some insight into the advantages and disadvantages of that architectural style. The other useful information from the CBML effort is the support for existing XML languages. Since the scenario conceptual model contained in the hyperscenario framework is based on different narrative domains, it stands to reason that there are (or will be) standard XML descriptions of those domains. A XML schema format, as opposed to a DTD-oriented grammar, could more readily merge existing XML specifications and minimize the need for mapping rules.

2.1.4 StoryML

Cynthia Kurtz at IBM T. J. Watson Research Center is the principal investigator for *Story Markup Language (StoryML)* research project [110]. Kurtz is a member of the Knowledge Socialization Group at IBM, whose purpose is to examine knowledge representation and sharing to support group collaboration . The StoryML specification language was developed to handle three areas of narrative: story composition, story organization, and group storytelling. StoryML was created by examining different story domains, such as: narratology, folklore, professional fiction writing, and case-based reasoning. The concepts were integrated into a XML schema representing metadata about stories and storytelling events. Kurtz derived the schema by exploring the six narrative domains she felt had the most potential for story metadata. The view evolved from determining commonalities and constructing a consensus of story elements. The StoryML approach differentiated between narrated events (the activities within the story) and narrative events (the process of storytelling). Story elements, such as, conflict, scripts, and consistent realities were combined into a simplified model that describe story metadata as form, function, and trace. *Story form* relates to the internal structural components of environment, character, plot, and narrative. *Story functions* are the relationships between characters in the story, between characters and their goals, and between individual stories. Function elicits the meaning, understanding, and convention of the story. The *story trace* is the context and the temporal development of the narrative. Kurtz lists a series of tool ideas proposed to manipulate StoryML; tools for authoring, modeling, organizing, searching, etc.

The StoryML project has much in common the hyperscenario framework, both in its narrative approach the language implementation choice. However, the story metadata model is based more on a categorization of story elements based form or function, as opposed to a conceptual model that defines direct relationships between them. StoryML was designed specifically for business stories in order to leverage narrative as a form of institutional knowledge. There is no discussion of how this narrative specification can be used to help decision-making. StoryML artifacts are suggested as representations of shared cognition, i.e., the memory and methods of an organization in story form.

2.2 Narrative Morphology

Scenarios are a form of narrative used for making decisions, essentially stories about systems or environments. For the remainder of this dissertation, the terms scenario and narrative will be used interchangeably. To address the first research question concerning the structure of scenarios, it is necessary to survey the diverse forms of narrative. This section describes a narrative morphology on which to base the scenario representation. A *morphology* literally means *study of forms.* The scenario ontology of the hyperscenario framework evolved by reviewing different forms of narrative and their utilization in problem-solving activities.

2.2.1 Narrative Context

Before beginning a narrative morphology, it is helpful to examine some narrative concepts. Figure 1 is a simple diagram to represent a taxonomy of terms in storytelling. The three divisions in the figure established by the dashed boundary lines represent the *story*, the *storyteller*, and the *audience.* Depending on the style of story, these separations may be arbitrary. A *story* is an account of incidents or events [126]. The *audience* is the body of listeners or spectators that experience a story. The *storyteller* acts as the interface between audience and story. The storyteller drives audience *perception. Perception* is the understanding or insight into the story. The dashed opening on the right-hand side of the figure is labeled *engagement. Engagement* is the amount of audience interaction. There are several levels of engagement. The lowest level of engagement is that of audience as passive receptacles of the story as it unfolds. The moderate level engagement is the *audience participation* style of narrative. This is embodied in dinner theater plays that require the audience to interact in order to move the story along. The widest level of engagement, immersion within the story, is the audience-as-storyteller as described in Murray's *Hamlet* *on the Holodeck* [133].

The circle on the figure represents the universe of the story; those *characters* whose interaction causes the narrative. These *characters* are the persons, places, and things whose actions cause *events*. Outside of this universe of characters and events are the *rules* that govern how they may act. Rules such as *rocks don't fly* and *people cannot walk through walls* set boundaries for what type of events might occur. Along with the rules, there are also the issues of *space* and *time*. When do things happen? Under what circumstances? Can the focus occur on two locations in the story at the same time? The answer to these questions along with the defining rules of interaction set the *scope*. Within that scope are the events that the storyteller relates to the audience. Depending upon how the story is told and the amount of engagement of the audience/storyteller there could be both anticipated and unanticipated events. There may be loopholes in the rules that allow events to occur that are not normally valid. The surprise results from these unexpected twists are usually from the perspective of the audience. Unanticipated events can also be from a participant's perspective, for example, a security guard watching Neo walk up a wall in the motion picture *The Matrix* [182]. The flow of the story, called its *continuity* , is how the events are organized and presented. The amount continuity can be altered is depicted as a dashed opening on the figure. The more control exerted by the storyteller/audience is increased, the larger the opening. The smallest opening is *keyhole* continuity, where the storyteller describes events one a time without modification of the established rules. At the other extreme, events can occur in any order and the rules may be modified to cause conflicts or make things more interesting. For example, the dungeon master in a game of *Dungeons and Dragons* can introduce characters and plot devices with which the participants may interact [33]. *Mechanics* are the *style* and format that the storyteller wishes to place upon the story. The children's story *Green eggs and ham* would have a different impact if the mechanics were those of Stephen King instead of Dr. Seuss. The tone, style, and flow of the story color how the audience may receive it.

Not all narrative forms include every element of the context. There are also other factors that influence narrative, such as the size of the audience, the number of storytellers,

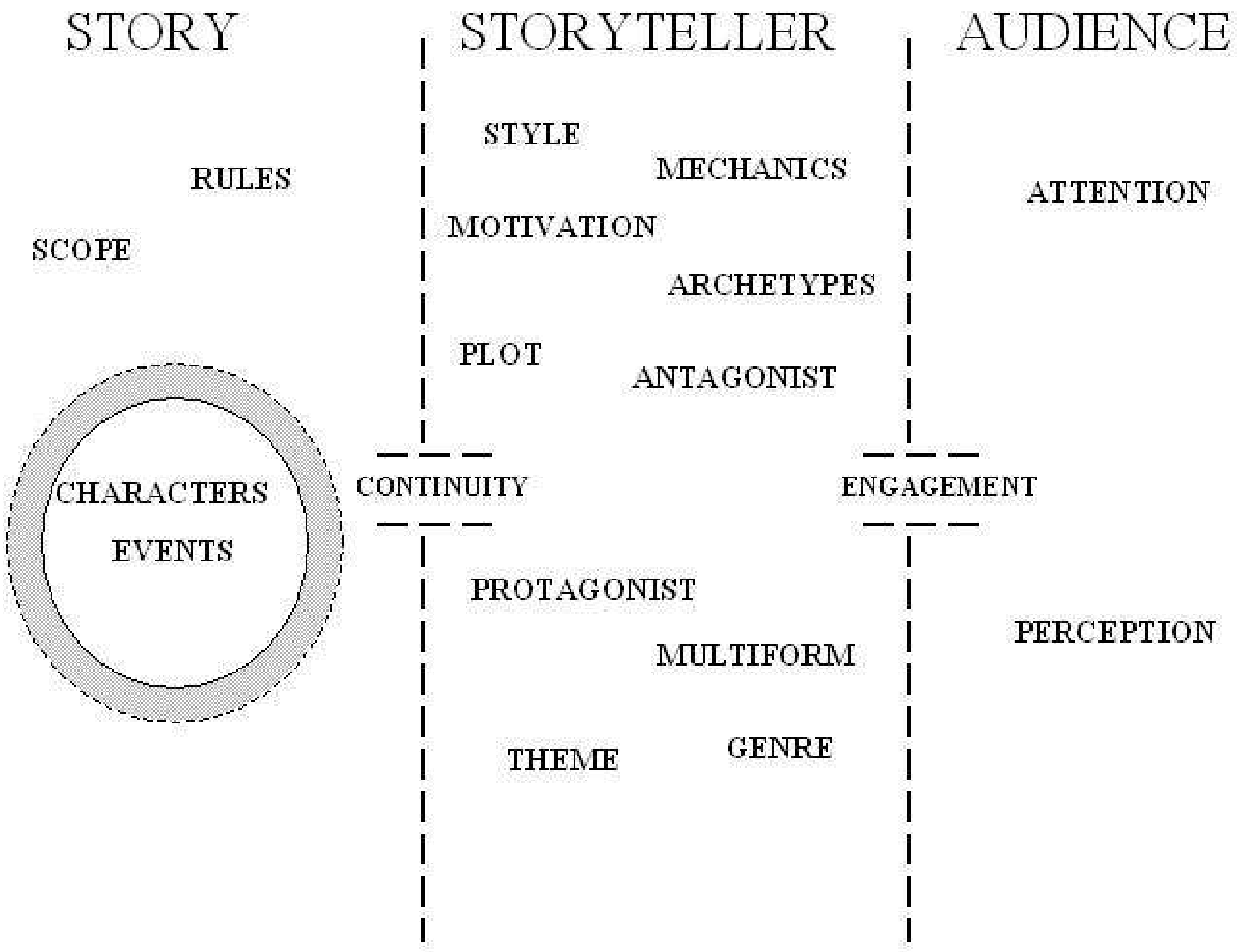


Figure 1: Storytelling Context

the media for storytelling (print, visual, auditory), the purpose of the story (educational, entertainment, informative), etc.

2.2.2 Literary Analysis and Criticism

Literature was an obvious domain for consideration, since the conventional concepts of a story are based on written descriptions of situations, locations, and events. Propp's *Morphology of a Folktale* was an effort by a 19th century literary scholar to create a hierarchy and structure of the standard folktale, as a basis for analysis [149]. Propp identified recurring patterns within the genre an built relationships between the patterns and the form of the folktales. Dallenbach, another literary critic, attempted to describe a specific narrative pattern, that of a play-within-a-play.[42] This type of story is a particularly useful storytelling device, when, for example, the narrator of a story is telling the tale as a series of flashbacks.

Joseph Campbell and Carol Pearson examined the concept of myths and storytelling as societal or cultural metaphors. Campbell's work is widely known as a treatise on the use of myth to explain the values and principles from a civilization. Examining how a group creates its mythos gives greater insight into what motivates their world. In *A Hero with a Thousand Faces* [20], Campbell talks about the goals and characteristics that we ascribe to heroes and how they are a microcosm of people in general. Pearson focuses on the *protagonist* of stories, defining a series of *archetypes* for categorizing goals and *motivations* [141]. A *protagonist* is the character that drives the plot and initiates the action. The *antagonist* is a character that attempts to block the goals of the protagonist. An *archetype* is defined by Pearson as "deep and abiding patterns in the human psyche that remain powerful and present over time". Pearson relates archetypes to phases of psychological development or personality type.

These literary approaches are relevant to the hyperscenario framework because they assign goals and motivations to particular roles and styles of stories. The goals and purposes of scenarios have to be able to relate to real life situations for them to be useful in a computational form.

2.2.3 Film Editing and Screenwriting

One catalyst for the hyperscenario framework was the view of a scenario as a document. Since scenarios are stories about a system that could take on alternate representations, was there an existing document type that exhibited similar characteristics? The screenplay documents produced for the film industry were a good example of this type of flexibility. In the film industry, the *screenplay* is a standard format for the document that describes a movie or a play [58]. The same screenplay can be used to create a storyboard, generate a soundtrack, describe costume requirements, forecast the budget, and outline casting criteria. There are several ways to represent the content from a screenplay: a storyboard of scenes, the soundtrack, the advertising trailers, etc. But in all cases, the structure of the original screenplay document remains consistent. A screenplay can describe movies in different genres, such as horror, comedy, and drama. A *genre* is a classification of story using the traditions under which the story is set as it relates to previously made stories [58].

The commonality among all screenplays is the narrative structure. There is a standard way of specifying actors, Settings, scenes, and actions. The salient parts of the documents are understood by the industry and relatively brief instructions can convey large amounts of information. The analogous document in theater or radio productions is a *script.* The standard script format is understood by the different theater perspectives.

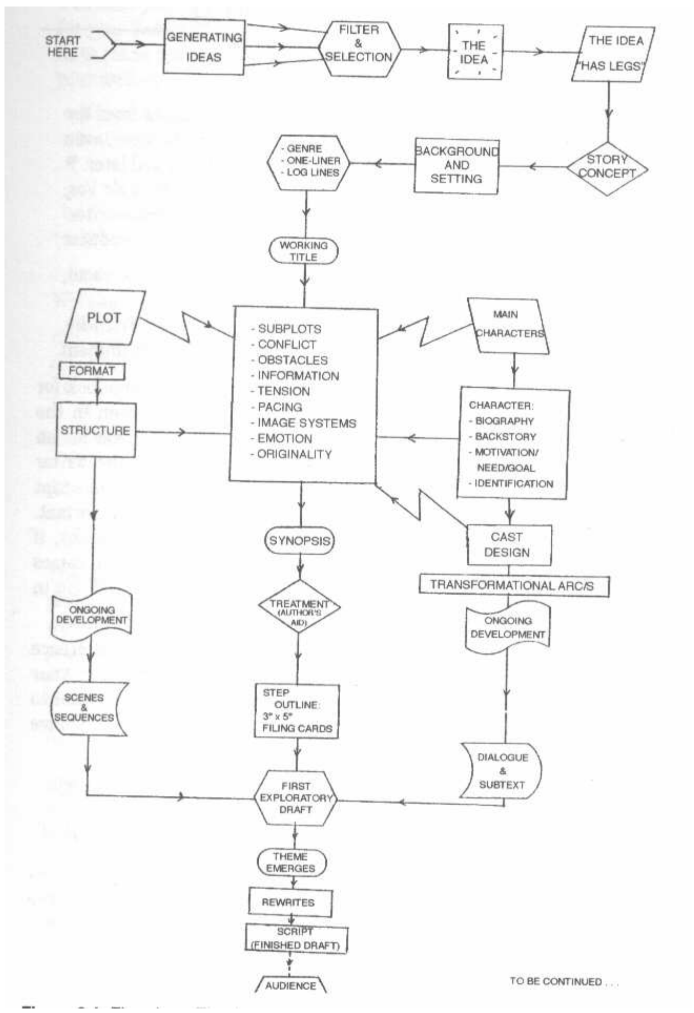


Figure 2: Screenwriting Process

The screenwriter does not directly translate a novel into a series of scenes, but gives some notion of a relationship between the story elements and an overall flow of action. Figure 2 is a flowchart of one version of the screenwriting process [58]. The same screenplay can be used to create a storyboard, generate a soundtrack, describe costume requirements, forecast the budget, and outline casting criteria. Higher level abstractions of the story, such as genre, plot, and theme are determined as the story is written. There can be multiple levels of a story, as in the *mise en abyme*, or play-within-a-play, described by Dallenbach [42]. A screenplay differs from a script in describing the visual aspect of the story from the point-of-view of the audience. A screenwriter is trying to elicit certain responses or target the story towards a certain perspective. The director has leeway in the set up of scenes and shooting individual frame, but a screenwriter has to interpret a particular view of a story. This is the reason that novels are converted to screenplay before filming begins.

The film's editor is responsible for *mise en scene*. *Mise En Scene* means the staging of scenes for film-making. Each scene is a movie is filmed from multiple angles or perspectives. The director and film editor then decide what form the story should take by choosing the shots that most effectively illustrate the story [43]. The choices can greatly enhance or diminish the impact of the story on the audience. This is the reason that many films have what is called a director's cut version of a film. The *director's cut* includes sequences and shots not chosen by the film editor in the theatrical release.

In her book *Computers as Theater,* Laurel speaks of the need for new design approaches for human-computer activity, using theatrical techniques as examples [113]. *Human-Computer Interaction (HCI)* revolves around the user interface. Direct manipulation systems that use the desktop metaphor allow users to view the computer as a collection of objects analogous to real world objects (files, folders, etc.). Laurel uses movie making and film editing ideas to support the notion of direct engagement systems that enable users to work directly in the activity of choice.

2.2.4 Cognitive Mental Models and Thought Experiments

Scenarios are analogous to the cognitive science notions of mental models and thought experiments. A *mental model* is an internal representation of the outside world [99]. When asked to *picture the earth*, most people have an internal visualization of a spinning, spherical object with patches of blue and brown representing oceans and land. A *thought experiment* is a hypothesis testing process where a person visualizes a sequence of events to determine a possible outcome [163]. If the internal visualization is a sufficiently reasonable model of the environment under which the events occur, the outcome will illustrate possible solutions to the problem. A scenario incorporates describing these internal models (with different levels of detail) with a hypothesis-testing focus.

Some influential cognitive scientists have elevated narrative to a central position in their models of explanation and inference [137] and in the application of these models to pedagogy and educational technology [108].

2.2.5 Hypertext

In the article *The Structure of Hypertext Activity* , Rosenberg introduces the concept of *actemes* and *episodes* [152]. An *acteme* is a low-level activity, such a following a link. A collection of actemes are termed an *episode*. These actemes relate to elements in the hyperscenario framework that support navigation between hypertext links. A *hypertext* is a representation of a story where interactive structural operations are intermingled with the text. With actemes, it is the structure of the language associates the actions with the episode. The semantics of actions depend upon the overall purpose and presentation form of the acteme.

2.2.6 Scenario-Based Design and Requirements Analysis

Scenarios have been used in software engineering to aid in decision making, comprehension, design, and training [23]. These scenarios are used largely as points of discussion in each of these cases. Building new scenarios or analyzing existing scenarios orient the discussion in collaborative activities and increase understanding in single user tasks [135].

During a 1993 workshop on user-oriented design representations, several leading practitioners in the fields of HCI, requirements engineering, *Computer Supported Cooperative Work (CSCW)*, and other disciplines within software engineering attempted to focus attention on the concept of scenario-based design [24]. The result was a collection of case studies, design perspectives, and approaches that illustrated the usefulness of scenarios to enhance traditional system design. Some of these studies on scenarios proposed methods for establishing scenarios as design artifacts to be used during system development. The output of the workshop included a list of roles of scenarios during system development:

- Requirements Analysis - Description of user wants and needs for system development
- User/Designer Communication - Scenarios as point-of-discussion between client and developer

- Design Rationale - Annotation of reasons behind design decisions
- Envisionment - Prototypical view of system before implementation
- Software Design - Description of the software modules and algorithms
- Implementation - Task lists to operationalize system actions
- Documentation and Training - Scenarios for end user instruction guides and maintenance trouble-shooting
- Evaluation - Test suites to exercise system functionality
- Abstraction - High level description of system behavior
- Team Building - Merging perspectives to create a coherent story from collaborative development

In each of these roles, scenarios are an informal approach to the refinement of software artifacts. Software prototyping, textual descriptions, storyboarding, user interface mockups, and simulations can all be considered forms of scenarios.

2.2.7 Use Case Scenarios

Scenarios are used so often in system development that they figure prominently in the notation for *Unified Modeling Language (UML).* Ivar Jacobsen, one of the originators of the UML notation, depicts use case scenarios as sequence diagrams representing collections of actions/event traces for a system under development [57].

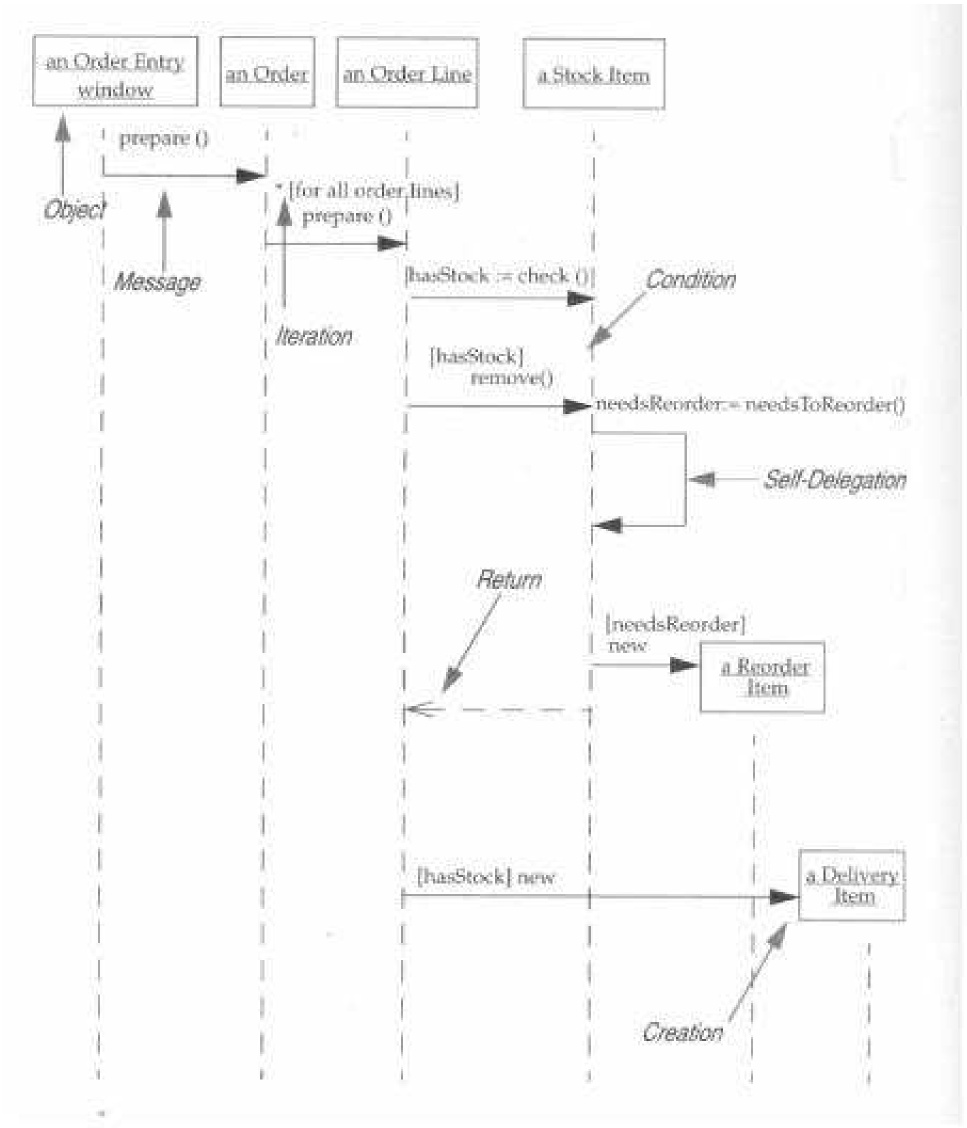


Figure 3: Customer Order Use Case Scenario

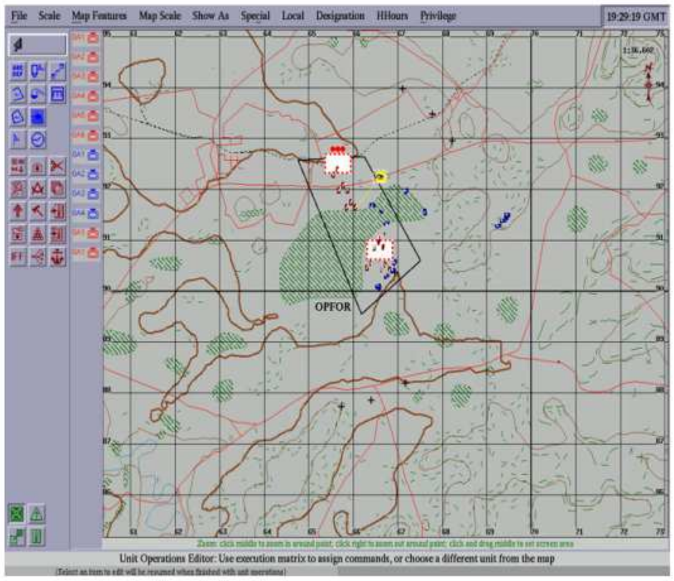
Figure 3 is a UML sequence diagram for a customer order processing system. A *sequence* diagram shows temporal information with each vertical line descending from system objects representing a timeline of events. This figure shows an *Order Entry Window*, an *Order*, an *Order Line*, and a *Stock Item* as the actors involved in the task of placing a customer order in a potential system. Following the time lines from top-to-bottom indicates the order of activities and the arrows on the diagram depict communication between actors.

2.2.8 Modeling & Simulation Environments

The *Department of Defense (DoD)* has focused much attention in recent years on *Modeling & Simulation (M&S)* environments for developing and analyzing the military systems [29]. Decreasing budgets, coupled with the necessity of being prepared for multi-faceted types of military activities, such as humanitarian interventions, joint peace keeping efforts, and *Missions-Other-Than-War (MOTW)* have created a need to prepare for unforeseen missions. There is also a requirement for the military to attempt to try out unfielded and evolving weapon systems and technologies to see how they might affect their *Tactics, Techniques, and Procedures (TTP)* [178]. This analysis is being done by means of simulation and modeling, from conventional (live action) wargaming to completely virtual simulations. *Constructive simulations*, the man-in-the-loop style of simulation, are being used by the military to determine the affect of new technologies on *Command, Control, Communications, Computers, and Intelligence (C4I).* These simulations take the form of war games in which participants are given assigned roles.

Modular Semi-Autonomous Forces (ModSAF) is an example of a simulation environment with virtual objects representing tanks, scout vehicles, artillery units, and other combat platforms (Figure 4). ModSAF simulates the hierarchy of military units and their associated behaviors, combat vehicles, and weapons systems. The participants run the simulation from static mockups of battlefield vehicles, using on-board computer systems with collaborative technologies (shared whiteboards, digital radios, etc.) to view the battlefield. During a simulation exercise, tank mock-ups and command vehicles communicate by radio according to standard protocols that govern actual mission interactions. This digital radio communication data, along with data streams from the simulation software are recorded as an event trace for subsequent analysis .

Capturing high-level, contextual information as well as sharing data objects can enhance simulation interoperability. The *High-Level Architecture (HLA)* consortium is a government-sponsored organization establish standards for simulations interoperability [181]. Specifically, the IEEE P1516.2 Standard for HLA *Object Model Templates (OMT)* prescribes a format to support the reuse of *Simulation Object Models (SOMs)* [180]. There is a major



Zoom: click middle to zoom in around point; click right to zoom out around point; click and drag middle to set screen area

Unit Operations Editor: Use execution matrix to assign commands, or choose a different unit from the map

(Select an item to edit will be resumed when finished with unit operations)

Figure 4: ModSAF Simulation Environment

effort in the simulations community, called the *Extensible Modeling and Simulations Framework (XMSF)* to propose a set of standards, processes, and practices to incorporate web services for the next generation M&S systems.

2.2.9 Decision Theory and Decision Analysis

Eighteenth century scholar Daniel Bernoulli describes decision making in terms of the *Expected Utility (EU)* theory. The EU concept states that when given a choice, people do not attempt to maximize expected value but the maximum expected utility. Since maximum expected utility is essentially the maximum subjective value, EU describes decision making in terms of value tradeoffs.

Another major decision theory was promoted by Thomas Bayes, also in the eighteenth century. The *Bayesian Theorem* of decision making states that decisions are made using probabilities of expected outcomes. Bayesian decision theory is a statistical approach for decision pattern classification using probabilities and the costs of error. Baye's formula states that the posterior probability of a decision's outcome is equal to the likelihood of the outcome times the prior probability divided by the evidence of the outcome [11].

Behavioral psychologists Yates and Estin define good decision making as follows: “A good decision is one that has few serious decision deficiencies” [190]. They list a series of common deficiencies that affect good decision making:

1. Aim Deficiency - *Aim deficiency* occurs when a decision fails to meet the decision-maker’s aims or objectives.

2. Instigating Need Deficiency - *Instigating Need Deficiency* occurs when decisions fail to ease the discomfort that required the decision-making.

3. Aggregrate Outcomes Deficiency - *Aggregrate outcomes deficiency* happens when all of a decision’s consequences are worse than if the decision-maker made no decision at all.

4. Competitor’s Deficiency - *Competitor’s deficiency* means there existed at least one alternative choice that is superior to decision made, with respect to aggregate outcomes.

5. Process Cost Deficiency - *Process cost deficiency* happens when there is an excessive expenditure of resources for a given decision.

Decision Analysis is a modeling approach that is a set of axioms, decision rules, and procedures for structuring the decision-making process [31]. Decision analysis is performed when the problems are complex, there are high stakes, there is no problem specific expert, or there is a need to justify decisions. Some of the factors that make decisions hard to make include conflicting objectives, intangibles, long time horizons, multiple decision makers, and the difficulty of identifying good alternatives. Decision analysis is *normative*, not *descriptive* [103]. *Normative* means that decision analysis is a systematic quantitative approach for better decision-making as opposed to a description of how unaided decisions are made.

2.2.10 Interactive narrative and role-playing games

Interactive narratives are immersive environments, where the participants experience the story first-hand and the users’ actions determine the plot. This level of engagement is a primary characteristic of *Role-Playing Games (RPGs)* where players act out the part of a particular role in a game, performing those behaviors associated with that character [124].

CDROM video games, such as *DOOM*, *Tomb Raider*, and *Myst* allow a player to immerse themselves into a scenario where the sequence of events are based upon player action [155]. At any point within these adventure games, the possible moves have been predetermined by the games designer . There is some leeway in how the player accomplishes a particular task, but for the most part the system can only respond in limited ways. Some of the most complex interaction in adventure games use the same interactive narrative technique as *Multiuser Dimensions (MUDs)* [10]. In the Early 80's, textbased MUDs such as the *Hitchhikers Guide to the Galaxy* [1] allowed for a large number of outcomes based on a handful of potential moves. MOO *(MUD Objectoriented)* environments like *SimCity* create virtual worlds where users can collaborate on design and share ideas by constructing scenarios. The MediaMOO project, which originated at the *Massachusetts Institute of Technology's (MIT's)* Media Lab, is a textbased, networked, virtual reality environment that was created to help media researchers. Like SimCity, MediaMOO is a constructionist environment, allowing participants to learn and be entertained by building personally meaningful artifacts [17]. The participants have flexibility in what they wish to create, constrained only by their creativity and the resources available in the software.

2.3 Summary

This chapter presented a literature review on narrative and highlights related work on scenario-oriented problem-solving. There are several research projects and software applications that use narrative for knowledge representation. The context for narrative revolves around the roles of the story, the storyteller, and the audience. The level of engagement and perception for the audience is guided by the mechanics and the style of the storyteller, along with background knowledge of the story environment, such as genre and theme. There are several forms of narrative, from film editing to use cases to mental models, that influence the choice of scenario elements for the hyperscenario framework.

Chapter 3

THE HYPERSCENARIO FRAMEWORK

See the little phrases go,

Watch their funny antics,

The men who make them wiggle so,

Are teachers of semantics

– *Frederick Winsor*

The narrative morphology discussed in the previous chapter establishes that people use stories for problem-solving. It is reasonable to conclude that a computer-readable scenario representation assists decision-making. This chapter describes the theoretical basis of this scenario research. It presents a formal description and definitions for the *Hyperscenario Framework.* A *Hyperscenario* is a specially formatted, computer-readable scenario that contains structures for navigation between scenario elements. The *Hyperscenario Framework* is a scenario ontology, scenario grammar, a markup language design, and the implementation of an XML-based scenario language.

The first section of this chapter discusses the motivation and introduces a scenario exemplar for illustrating the framework. The second section presents the scenario ontology, scenario construction, querying/analysis operations, and some of the essential operations for telling stories from the model. Section three discusses the structure of the scenario grammar based on the ontology. A brief overview of *Extensible Markup Language (*XML) is presented in section four as the specific choice for creating the scenario language. The syntax for the implemented language, *Scenario Markup Language (*SCML*)* is then introduced. The final section of the chapter suggests some general heuristics for finding scenarios within a narrative domain for representation in SCML, along with a discussion of the requirements, limitations and disadvantages of the hyperscenario framework.

Figure 5: Automated Shuttle System

3.1 Motivation

An *ontology* is a formal explicit description of concepts in a domain of discourse, the properties of each of those concepts and their attributes, and restrictions on those properties [37]. An ontology is not just a conceptual model, but also includes rules and policies that describes elements within the model. It is helpful to have a running example of a scenario environment to motivate the identified concepts in the scenario ontology.

The third international Workshop on Scenarios and State Machines (SCESM04), held in Edinburgh in conjunction with the International Conference for Software Engineering 2004 (ICSE04), supplied scenario case studies for use by participants to evaluate their scenario research. One of those case studies, involving the simulation of an automated rail shuttle system, is the running example for this chapter on the hyperscenario framework.

A rail-based shuttle system is being developed under a research project at the University of Paderborn[1] . Figure 5 is a snapshot[2] from an on-line web camera showing a portion of the actual rail shuttle system that has a total length of approximately 600 meters. The system is intended to enable individual traffic of passengers and materials using autonomously acting rail shuttles. A simplified scenario for the shuttle control software and a simulation environment have been developed by Dr. Wilhelm Schäfer and Dr. Ekkart Kindler of the Software Engineering Group at the University of Paderborn [106]. Schäfer and Kindler taught an undergraduate software engineering course which required teams of students to re-engineer and improve the railway simulation software. The students' primary goal was to implement an intelligent shuttle and add further environment components or modify existing ones [61].

In order to make it accessible to a broader audience, the shuttle system has been documented in the form of a case study. The case study was made available in the call for participation of the SCESM04 conference. The following paragraphs are directly extracted from the shuttle system case study[107]. It is necessary to include the exact wording of the case study in order to derive the scenario elements from an actual domain document.

> *Consider a railway. The railway consists of interconnected stations. Shuttles bid for orders to transport passengers between certain stations. Successful completion of an order results in a monetary reward for the shuttle involved. In case an order has not been completed in a given amount of time, a penalty is incurred. New orders are made known to all shuttles, thus all shuttles can make an offer. The shuttle with the best, i.e. lowest offer will receive the assignment. Using the tracks will incur a toll, depending on the distance covered. Maintenance of the shuttles is possible at any station and will cost both time and money.*
>
> *The railway network consists of stations, track and switches. Tracks can be traveled upon in one direction only. A switch is configured as a converging or branching junction depending on the directions of the adjoining tracks. A section of tracks consists of any number of connected tracks and switches. A section of*

[1] http://www-nbp.upb.de/en/index.html
[2] scale of 1 : 2.5

tracks between stations is called a connection. The direction of a connection is determined by the direction of its constituent sections of tracks. A connection can only be traversed in the predefined direction and changing the direction while underway is not possible. Connections can share sections of tracks and switches, while there must only be one connection between two stations in one direction.

Any number of shuttles can be present at a station at the same time. The duration of a shuttles stay at a station is not considered maintenance time. Maintenance must be explicitly scheduled.

Connections between stations can be temporarily disrupted. Shuttles currently traveling on a section of tracks at the time of the disruption are not affected by it, and will be able to complete their journey. Shuttles at stations will not be able to use the connection. They can however use a different route. All shuttles will be informed of the disruption and its duration.

Orders are made known to all shuttles by a broker. An order defines start and destination stations as well as the allowed time for completion. The deadline is derived from the time of acceptance of an order and the predefined processing time, which begins at the time of acceptance. Additionally, an order has a certain size, namely the number of people wishing to travel. Orders will be paid for by the passengers either by credit card or invoice.

Order assignment follows a strict pattern. Firstly, all shuttles are informed of the new order. Within a certain period of time, any shuttle can make an offer, which defines the payment it will receive after successful completion of that order. The shuttle having made the lowest offer will receive the assignment. In the event of two equal offers, the assignment will go to the shuttle that first made the offer.

Order processing is handled by the shuttles. Every shuttle can transport passengers up to a maximum capacity determined at the start of the simulation. This means that a shuttle can transport more than one order at the same time, as long as the orders do not exceed the maximum capacity. The number of orders assigned - but not necessarily loaded - to a shuttle at any given time, is not *limited. To complete an order a shuttle has to travel to the start station, load the order and then proceed to the destination station to unload. This sequence must be completed in a predefined time span; otherwise a penalty will be levied.*

Order-processing begins with the loading at the start station and ends with unloading at the destination. Loading or unloading at other stations is not permitted.

A shuttle traveling on a section of tracks can neither change direction nor choose another destination. A travel decision is only possible at a station before the journey has begun.

The Hyperscenario Framework discussed in this chapter is meaningful to three types of users corresponding to the storytelling context discussed in Chapter two: domain experts, storytellers, and audience. Each of these users are equivalent to one of the partitions of the storytelling context as shown in Figure 1 on page 14. The domain expert serves the role of the story portion of the storytelling context. It is the responsibility of the domain expert to determine the rules for the kinds of stories that are to be told in the domain. The domain experts determine the events, characters, actions, and the vocabulary that are meaningful. The storyteller's position in the context is as described in Figure 1–the responsibility for motivating, stylizing, determining genre, applying theme, and creating plots for stories. The storyteller takes those elements identified by the domain expert and creates specific stories in the domain. An example of the relationship between domain expert and storyteller would be the canon of Greek mythology (domain expert) and Homer (the storyteller). The final user of the Hyperscenario Framework is the audience who applies the perception, attention, and level of engagement to experience a story. Depending upon the level of engagement, an audience may be described as a participant that interacts with the story. Similar to the partitioning of the storytelling context, the boundaries between the Hyperscenario Framework users could be transparent. The domain expert who identifies the story pieces could be the storyteller who puts the pieces together into a story and, finally, the audience/participant that experiences the story.

There is a fourth category of user that will take advantage of the Hyperscenario Framework: the hyperscenario storyteller. The hyperscenario storyteller not only creates the plot of a story, but inserts structures in the story to support navigating the story in different ways.

These four kinds of users and the shuttle system exemplar are used throughout this chapter to motivate the scenario ontology, the scenario grammar, language design and implementation, and to illustrate the usefulness of the framework.

3.2 Scenario Concepts, Modeling, and Representation

The purpose of the Hyperscenario Framework is to enable scenario applications that help decision making. By putting scenarios into computer-readable form, the stories can be manipulated and queried to answer "what if" questions and make decisions. For the shuttle system, there are decisions about the number of shuttles required for the system, the most efficient route for transporting passengers, railway scheduling, ordering processing, and other choices that are assisted by examining scenarios about system usage.

To make decisions about the shuttle system or any environment that can take advantage of the Hyperscenario Framework, it is important to develop an ontology for scenarios. It is the scenario ontology that is useful to the domain expert. Developing the concepts and rules for scenarios in the ontology will inform the domain expert what pieces to look for that will be used for stories in a domain.

This section describes the details of the scenario ontology. Section 3.2.1 describes the structural model that identifies the concepts in the scenario ontology and their relationships. Section 3.2.2 describes techniques for connecting the conceptual elements together and navigating within a scenario. These operations become the basis for scenario construction and storytelling operations. Section 3.2.3 discusses some of the queries and decision-making tasks that are supported with the scenario ontology.

3.2.1 Scenario Structural Model

The first step in developing a scenario ontology is to identify the concepts that make up the scenario domain. Each person, place, or thing is an entity in the scenario ontology. These entities are then expressed as classes in a conceptual model showing their relationships and any implicit rules in the domain.

Figure 6 is a scenario conceptual model depicted in the UML notation. *Unified Modeling Language (UML)* was developed as a way to represent objects and classes in object-oriented designs [57]. The choice of UML modeling notation is purposeful: each scenario component is a class or category of elements. For the sake of this discussion, it is necessary to know only a few UML symbols to interpret the model.

- ▤ ***Class:*** Each rectangle in the diagram represents a class of objects. The name of the class appears in the top section of each rectangle. The two other sections of the rectangle list attributes and operations, respectively.

- — ***Association:*** A line between rectangles represents an association between classes. Each line coming out of a class can be numbered with a range of values representing the multiplicity of the association. The range is of the form ***m..n***, where ***m*** is the minimum number and ***n*** is the maximum number allowed. If there are no numbers on an association, the multiplicity is one-to-one. Associations between elements can be named, usually with a descriptive name. There may be an arrowhead on an association that shows the direction of navigation. The absence of an arrowhead denotes an association that is bidirectional. Associations can be *association classes* that themselves contain attributes and operations.

- ◇ ***Aggregation:*** An open diamond means the adjacent class is a collection of the elements on the other end of the association. This is equivalent to the *hasA* relationship from entity-relationship diagramming [30].

- △ ***Inheritance:*** The triangle represents inheritance or specialization. The relationship between the class touching the triangle and the class on the other end of the association is a *kindOf* association.

In the conceptual model of Figure 6, there are several associations between container

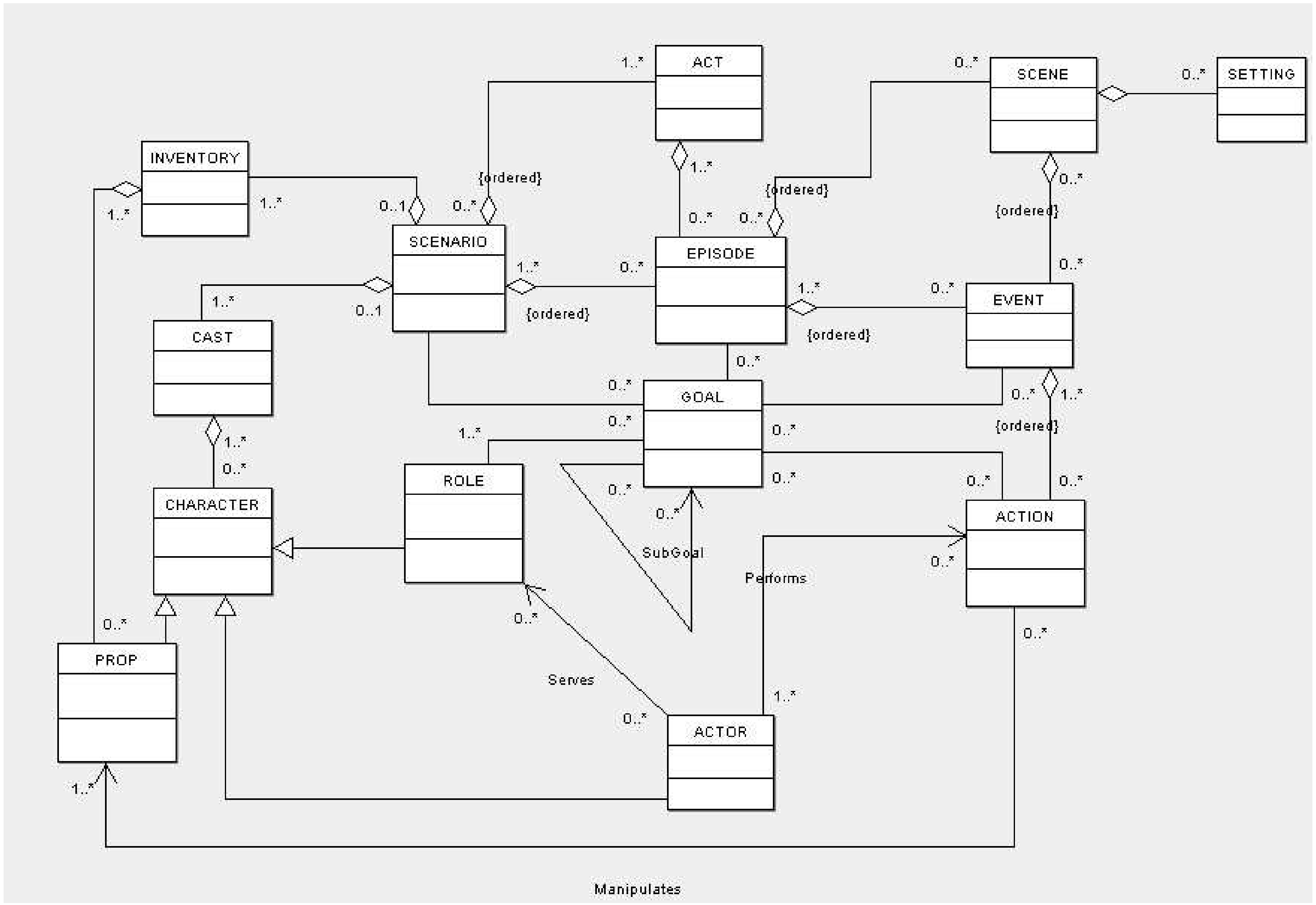


Figure 6: Scenario Conceptual Model

elements and aggregated elements labeled with the UML keyword `{ordered}`. If the keyword `{ordered}` is present on an association, the collection is in a specific sequence. To simplify the discussion, any element that is organizable into a specific sequence based on the `{ordered}` notation is referred to as a *step* for the remainder of this chapter. It is the domain experts that determines the sequence and the type of sequencing for ordering the steps of a scenario. All of the rules for the scenario classes that are defined in the next sections are either implicitly represented in the diagram and/or explicitly defined by the text.

A *Scenario* is a sequence of *Episodes.* A *Scenario* has at least one *Episode.* A *Scenario* has associated with it one and only one *Goal.* A *Scenario* may have a *Cast* associated with it. If there is a *Cast,* there is only one *Cast.* A *Scenario* may have an *Inventory* associated with it. If there is an *Inventory*, there is only one *Inventory.* The *Episodes* that are in a *Scenario* may be grouped into a sequence of *Acts.* If there are *Acts*, there are one or more *Acts.* The redundant path in the UML diagram allows for *Episodes* to stand alone or be grouped into *Acts* in a *Scenario.*

An *Inventory* is a collection of one or more *Props.* An *Inventory* has at least one *Prop.* An *Inventory* can be associated with more than one *Scenario.*

A *Cast* is a collection of one or more *Characters.* A *Cast* has at least one *Character.* A *Cast* can be associated with more than one *Scenario.*

A *Character* is defined in this model as any entity that is part of a story. A *Character* can be associated with zero or more *Cast*s. It is possible for a *Character* to exist without being associated with a *Cast.* The *Character* class in the UML model is a superclass with specialized scenario elements derived from it. There are three kinds of *Characters* that are important for the scenario ontology: *Characters* that do things, *Characters* that have things done to them, and *Characters* that have *Goals.*

Actor is a subclass of *Character.* An *Actor* is a *Character* that does things. There is a named association between *Actor* and *Action* called *Performs.* An *Actor* is a *Character* that *Performs* one or more *Actions.* An *Actor Performs* at least one *Action.* There is a named association between *Actor* and *Role* called *Serves.* An *Actor Serves* zero or more

Roles. For example, the designer, developer, technical writer, test and evaluation staff, and customer are different roles that may be associated with the same person. An *Actor* does not have to *Serve* a *Role.*

Prop is a subclass of *Character.* A *Prop* is a *Character* that has things done to it. A *Prop* is associated with an *Action* through the *Manipulates* association. A *Prop* is a *Character* that is manipulated by an *Action.* A *Prop* must be *Manipulated* by at least one *Action.* A *Prop* can be associated with more that one *Inventory.* It is possible for a *Prop* to exist without being associated with an *Inventory.*

Role is a subclass of *Character. A Roles* is a *Character* that has one or more *Goals* associated with it. A *Role* must have at least one *Goal.* There is a named association between *Role* and *Actor* called *Serves.* It is the *Role* that an *Actor Serves* that establishes the *Actor*'s *Goal* in the story. *A Role* may be associated with zero or more *Actor*s. It is possible to have a *Role* that has no associated *Actor.*

Act is a sequence of *Episodes.* An *Act* must have at least one *Episode. Acts* are necessary if there is a dramatic structure to the narrative. For example, the standard three-act structure for a screenplay includes the setup, conflict, and resolution [58]. *Acts* give additional meaning to groupings of *Episodes.* An *Act* can be associated with one or more *Scenario*s. An *Act* is associated with at least one *Scenario.* If there are *Acts* in a *Scenario*, all of the *Scenario's Episodes* are contained within those *Acts* in the same *Episode* sequence.

Episode is a sequence of *Events.* An *Episode* has at least one *Event.* An *Episode* has associated with it one and only one *Goal.* The *Events* in an *Episode* are grouped based on their support for the *Episode*'s *Goal.* Prior to constructing an *Episode*, a domain expert has to have defined what pattern of available *Events* supports the *Goal* of the *Episode.* The *Events* that are in an *Episode* may be grouped into a sequence of *Scenes.* An *Episode* does not have to have *Scenes.* The redundant path in the UML diagram allows for *Events* to stand alone or be grouped into *Scenes* in an *Episode.* An *Episode* can be associated with more than one *Scenario.* It is possible for an *Episode* to exist without being associated with a specific *Scenario.* An *Episode* can be associated with more than one *Act.* An *Episode* does not have to be grouped within an *Act.*

A *Scene* is a grouping that has a *Setting* and contains a sequence of *Events*. The possible *Settings* of *Scenes* are identified by the domain expert. A *Scene* contains zero or more *Events*. It is possible for a *Scene* to contain no *Events*. A *Scene* has associated with it one and only one *Setting*. A *Scene* may be associated with zero or more *Episodes*. If there are *Scenes* in an *Episode*, all of the *Episode's Events* are contained within those *Scenes* in the same *Event* sequence. It is possible for a *Scene* to exist without being associated with a specific *Episode*.

A *Setting* is a location, either temporal, physical, or abstract. A *Setting* may be associated with zero or more *Scenes*. It is possible for a *Setting* to exist without being associated with a specific *Scene*.

An *Event* is a sequence of *Actions*. The purpose of an *Event* is to organize meaningful collections of *Actions*. The rules and properties that make an ***Event*** and its corresponding *Action* grouping meaningful are defined by the domain expert. An *Event* must have at least one *Action*. An *Event* may have zero or more *Goals* associated with it. An *Event* does not have to have a *Goal*. If an *Event* has a *Goal* and is associated with an *Episode*, the *Event's Goal* is a subgoal of the *Episode's Goal*. An *Event* can be grouped within zero or more *Episodes*. It is possible for an *Event* to exist without being associated with a specific *Episode*. *Events* may be organized within an *Episode* by *Scene*. It is possible for an *Event* to exist without being associated with a specific *Scene*.

A fundamental building block of a *Scenario* is the *Action*. It is not a collection of any other scenario classes. *Actions* are anything that happens in the context of the story. An *Action* may consist of characters communicating, the physical movement of an object, or some other change of state in a *Scenario*. An *Action* is grouped within zero or more *Events*. It is possible for an *Action* to exist without being associated with a specific *Event*. An *Action* has zero or more *Goals* associated with it. An *Action* does not have to have a *Goal*. If an *Action* has a *Goal* and is associated with an *Event*, the *Action's Goal* is a subgoal of the *Event's Goal*. An *Action* is *Performed* by zero or more *Actors*. If an *Action* has a *Goal* and is associated with an *Actor*, the *Action's Goal* is a subgoal of the *Goal* of the *Actor's Role*. It is possible for an *Action* to exist without being associated with a specific *Actor*. An Action *Manipulates* zero or more *Props*. It is possible for an *Action* to exist without being associated with a specific *Prop.*

Several of the scenario elements are associated with a *Goal.* A *Goal* is an objective; a desired state to be achieved. It is possible for a *Goal* to exist without being associated with a *Scenario*, *Episode*, *Event*, *Role*, or *Action.* The *Goals* of the scenario elements in a complete scenario form a goal hierarchy with the *Scenario Goal* as the root element (Figure 7). Each *Episode* class within the *Scenario* has a *Goal* that supports the objectives of the *Scenario's Goal.* The *Events* have *Goals* that supports the objectives of the *Episode's Goal* above them. All the *Actions* within an *Event* are performed by an *Actor.* Since *Actors* do not have *Goals*, it is the *Role* that an *Actor* serves that is listed in the goal hierarchy. These *Roles* have *Goals* that support the objectives for the *Events' Goals* in which they occur. The *Actions* that are performed by the *Actors* in those *Roles* support the objectives of the *Goals* of the *Roles.*

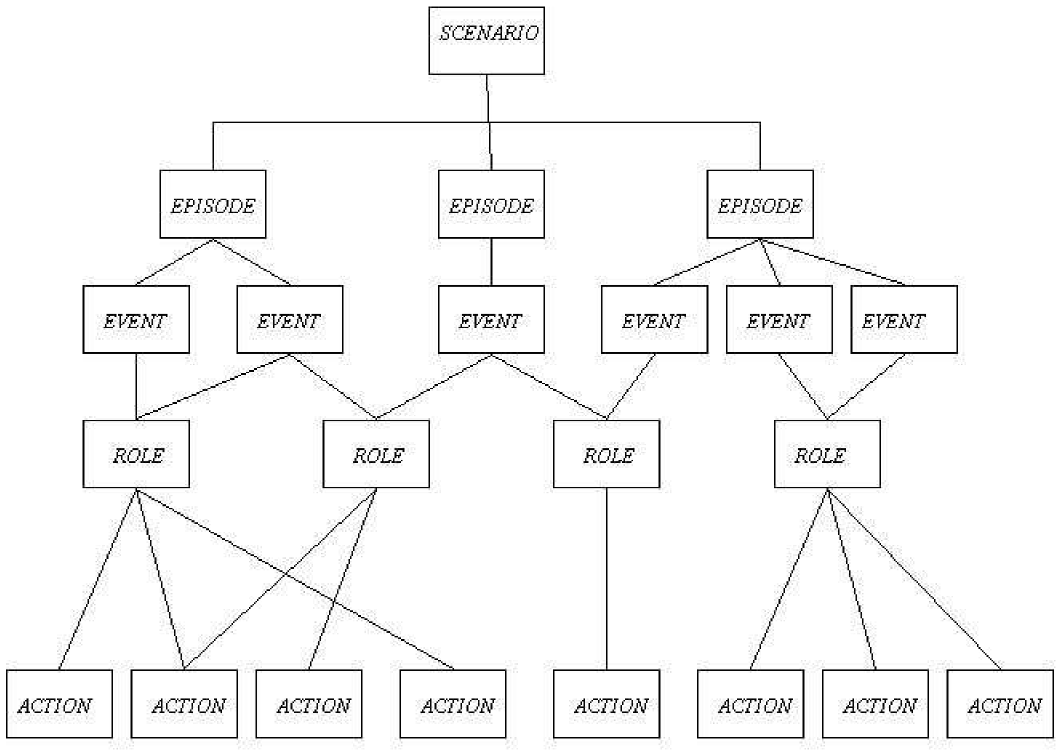


Figure 7: Scenario Goal Hierarchy

It is important to note that *Cast* and *Inventory* are header information for a scenario, not actually within the scenario. They are groupings of the *Characters* and *Props* in a scenario in a useful way to support subsequent analysis. It is difficult to depict this notion of header information in a UML diagram. This is the reason *Cast* and *Inventory* may not be present in a scenario. All of the *Characters* and *Props* that are listed in the *Cast* and *Inventory* can be derived by analyzing the scenario. Having the *Cast* and *Inventory* elements serve the same purpose as the closing credits at the end of a movie; it allows the viewers to know the entities in the story all in one place. Since *Props* are also *Characters*, they may be listed in both the *Inventory* and the *Cast*.

The Shuttle system example can be used to illustrate concrete instances of the elements from the conceptual model. Creating these instances are the responsibility of the domain expert. The following sections show an informal method for determining scenario elements from a textual description. The ordering of the activities is a useful way of constructing scenarios from this type of information. Since *Events*, *Scenes*, *Episodes*, *Acts*, and *Scenarios* are sequences of specific scenario elements, it is more illustrative to give a specific shuttle hyperscenario example of those scenario elements after introducing dynamic structures for connecting those sequences. Section 3.2.2 presents the dynamic model and uses the shuttle example to describe the construction of a complete hyperscenario.

Table 1: Shuttle System Character List

CHARACTER
Station
Shuttle
Order
Passenger
Section
Track
Toll
Switch
Time
Money
Broker Agent
Banking Agent
Shuttle Agent
Topology Agent
Disabling Agent
Simulator
Invoice

3.2.1.1 Find Characters

The Characters in a scenario are entities that might occur in the domain. The Characters can be determined by looking for people, places, or things that are within the natural language text description. Table 1 is an initial listing of Characters derived from the shuttle description information.

Table 2: Shuttle System Action and Prop List

ACTION	PROP
Configure	Switch
Transport	Passenger
Bid	Order
Levy	Penalty
Incur	Penalty
Change	Direction
Move Forward, Back	Station
Schedule	Maintenance
Disrupt	Connection
Assign	Order
Load, Unload	Order, Passenger
Decide/Choose	Route, Station
Perform	Maintenance
Create, Delete	Order
Check	Requirements
Calculate	Order
Make	Order
Evaluate	Order

3.2.1.2 Find Actions

The simplest method for determining Actions in a domain is searching for verbs in the textual description. It is also helpful to analyze any command list that might be included in the domain documentation. In Table 2, the second column lists the target of the Actions which are the props manipulated by those actions. There are Actions, such as `Decide/Choose` in the table, that may be synonymous and essentially equivalent in the domain.

Props are defined as the target of Actions. The information in column two of Table 2 lists the objects of the verb/actions as the primary candidates for props. There are additional Props that can be determined by analyzing the Character list and description document for passive entities that receive the results of Actions.

3.2.1.3 Identify Settings

The Settings are locations where Characters exist and Actions happen in the domain (Table 3).

Table 3: Shuttle System Settings

SETTING
Track
Station
Shuttle
Simulation

3.2.1.4 Identify Actors

Actors are those characters that perform actions. Any character that performs an action from the list is a potential actor in a a scenario. Table 4 is the list of actors for the shuttle system.

Table 4: Shuttle System Actor List

ACTOR
Shuttle
Passenger
Track
Switch
Broker Agent
Banking Agent
Shuttle Agent
Topology Agent
Disabling Agent
Simulator

3.2.1.5 Identify Roles

Roles are characters that have goals associated with them. These goals are associated with the actors that serve in those roles. A good rule-of-thumb is that if several characters can be associated with a goal, they are serving a role. For example, in Table 5, the role of a `Reward` can be served by the characters `Time` or `Money`.

Table 5: Shuttle System Roles

ROLE	**GOAL(s)**
Disabling Agent	Disabling connection between stations
Topology Agent	Send Topology data to shuttles
Banking Agent	Paying the shuttle
Broker Agent	Control Orders
Shuttle	Complete Orders, Transport Passengers
Expense	Expenditure for Transportation
Toll	Transport Fee
Maintenance	Repair and Maintain Shuttle
Reward	Compensate for Efficient Services
Connection	Connect multiple sections of track
Junction	Connect multiple path
Income	Revenue from Services
Payment	Cost of Services

3.2.2 Dynamic Model and Scenario Construction

The narrative morphology described in chapter two is also the basis for modeling dynamic scenario navigation. The term *dynamic* is used because there are scenario applications, such as browsing, simulation, walkthroughs and run-time analysis where it is useful to move through a scenario. The main goal of the hyperscenario storyteller is to insert structures in the scenario for these alternate forms of movement. The utility of the Hyperscenario Framework is in its use for storytelling and *what-if* manipulation. Moving through the scenario in alternate ways corresponds to different kinds of storytelling. The hyperscenario storyteller can use the constructs identified in the dynamic model along with the concrete instances of scenario elements defined by the domain expert to create a hyperscenario.

Storytelling can be thought of as a form of purposeful navigation through a set of events

[65]. Whether the purpose is comprehension, learning, entertainment, or speculation about an imagined environment, it is the task of the storyteller to weave these event chunks into a meaningful whole. In film-making, it is the job of the film editor to perform this task. The process of film-editing involves taking pieces of film that were created out of sequence and from various viewpoints and merging them into a coherent story [43].

Film-editing terminology is useful in describing the non-linear movement possible within a scenario. For example, the concept of a *Scene* is important in the transformation from plot description to implemented narrative. A *Scene* is a portion of film that takes place in a particular location. In a scenario, the location that distinguishes a scene need not be a physical place, it could be chronological or abstract. Examples of chronological location are "morning" or "evening". An example abstract location is "inside the database". There are many examples in literature and film-making where non-chronological presentation of a story enhances its impact. The movies *Pulp Fiction* and *Memento* are both made more memorable because their stories are not presented from beginning to end.

The dynamic model that is described in this section outlines the different ways of navigating a scenario. There are three methods for scenario navigation: links, portals, and shifts. A *link* is a connection from an action to an action. Links are used to connect steps, which can be re-organized by following paths through the links. A *portal* is an anchor/position for entering or leaving a scenario. *Shifts* are methods for accessing information external to a scenario, including even another scenario. The methods for movement within a scenario are links; portals and shifts are methods for moving around externally to a scenario. Just as a film-editor takes sequences of film created by the director and organizes them, the hyperscenario storyteller takes a scenario and uses links, portals, and shifts to transform it into a hyperscenario.

Figure 8 illustrates the navigation methods between scenario elements. The smaller diagram in the left side of the figure depicts different kinds of links. All are links because they connect actions; it is the purpose and the method for connecting the actions that establishes the different types.

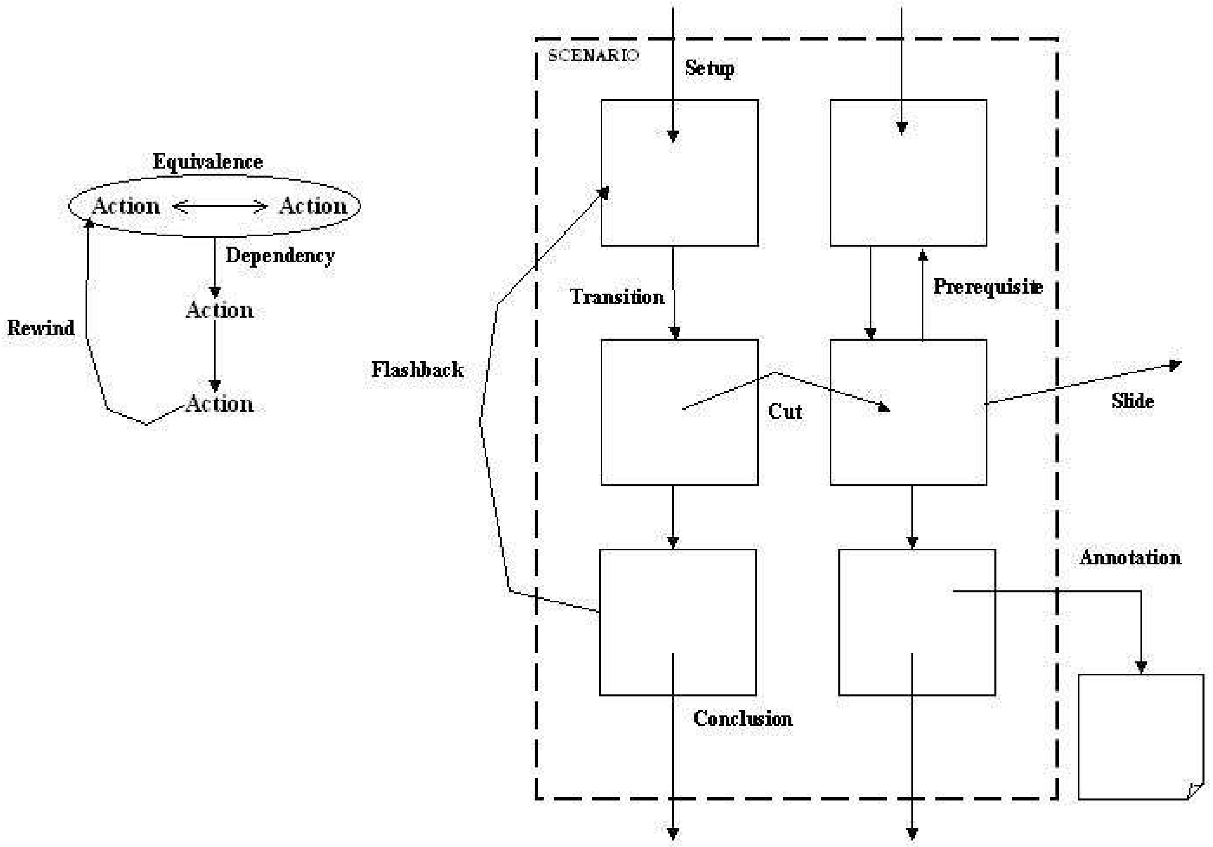


Figure 8: Hyperscenario Navigation Model

A *dependency* link between actions establishes an ordering between them. The dependency link connects two actions and denotes which has to happen first. All links are connections between actions, but the dependency link is the only type of link that establishes that one action is dependent on another. The action at the source of the dependency link has to occur before the action at the destination of the link.

The actions enclosed by an oval in the smaller diagram in Figure 8 are equivalent to each other. A *equivalence* link establishes a class or set of actions that are synonymous. Actions connected with an equivalence link have the same meaning in their domain and are interchangeable. The choice of any action within the set has the same effect on the flow of the scenario. For example, there may be a domain where the list of actions contain the terms “throw” and “toss”. Both actions have been designated by domain experts as present in the vocabulary of the domain, but their meanings are essentially the same. The hyperscenario storyteller designates these terms as equivalent and associate them with an equivalence link, which allows them to be interchangeable. Replacing one with the other does not affect the meaning of the scenario.

A *rewind* link allows for backing up within a sequence of actions. A rewind link is installed by the hyperscenario storyteller to allow backward movement in a scenario. The link connects in reverse, from the tail of a sequence of actions to the head of the sequence of actions, without going through the individual actions in between. The sequence can then be repeated by following the rewind link to the beginning of a sequence.

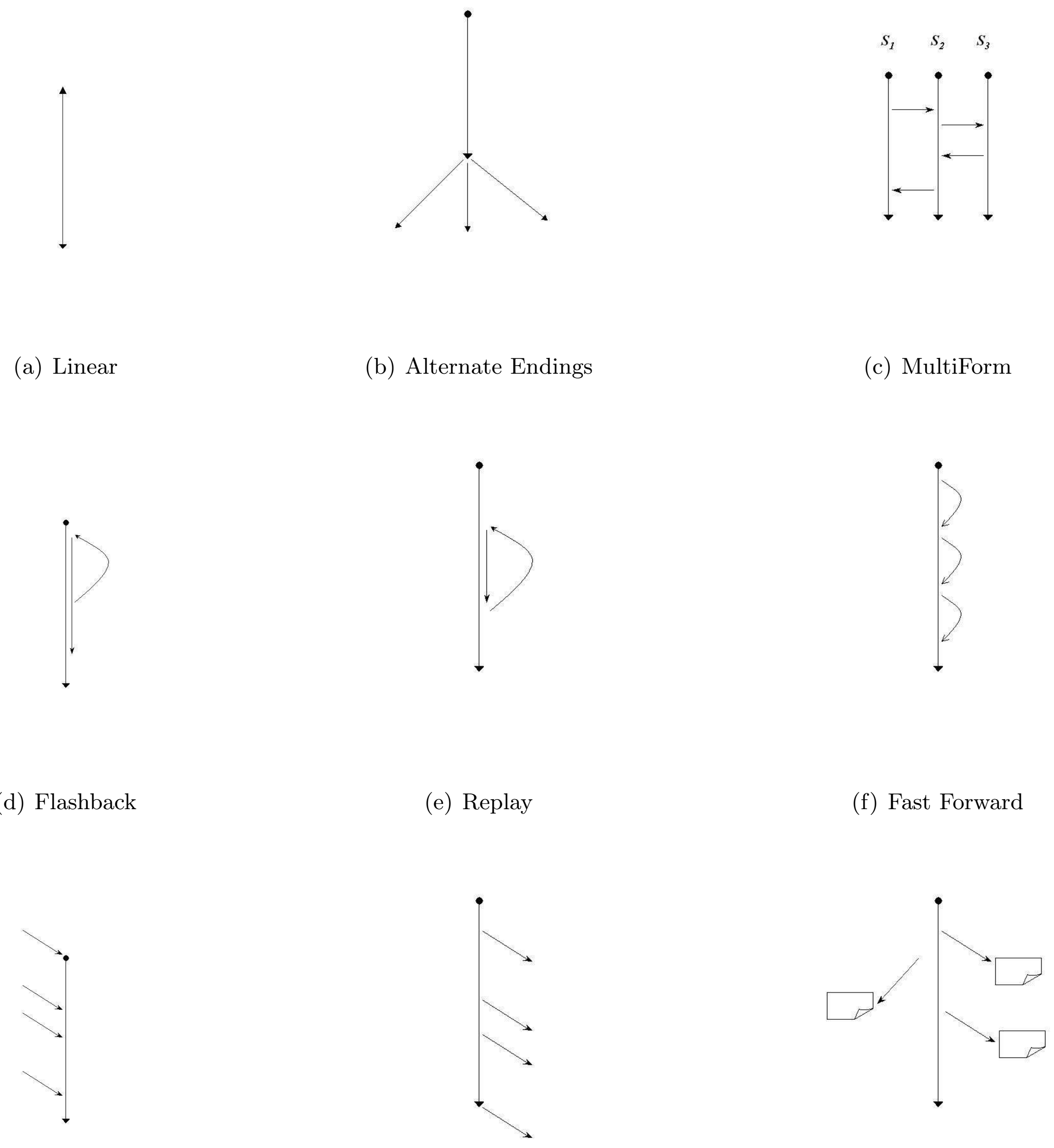
The larger diagram on the right of Figure 8 show navigation methods within a scenario. The dashed outside box represents a single scenario, with the internal boxes depicting individual episodes within the scenario. The two columns of boxes in the diagram represent different viewpoints of the same scenario. A viewpoint in this context is the perspective of a character in the scenario. A horizontal pair of boxes represents the same episode from different viewpoints. For example, given two characters A and B in the scenario, one box represents an episode from the perspective of character A while the corresponding box is the episode from the perspective of character B. Changing viewpoints can be done using a *Cut*. A *Cut* is a link from one actor's perspective to another. A cut is a link because it connects from an action to an action. The actions that are connected by a cut are from two different actors. A cut connects from the action of an actor in an episode to the action of an another actor in the same episode. Cuts can be used to allow characters to interact and to elicit some level of engagement from the audience. Effective use of cuts decreases the amount of exposition or explicit narration.

Flashbacks, transitions, and prerequisites are links intended to connect episodes. A *flashback* link is a connection from an episode to a preceding episode to review background information that is present in the scenario. A *prerequisite* link connects an episode to previous episode based on the dependencies between actions in the episodes, unlike flashbacks which are based on supplying information. *Prerequisite* links also differ from flashback links in that the episodes are always adjacent to each other in sequence. Flashbacks can be one or more hops previous to the episode being examined. The *Transition* links connect episodes going forward in sequence using action dependencies. A transition link establishes the dependency between episodes, unlike a dependency link, which establishes a dependency between actions. For example, the forest episode has to occur before the grandma's house episode in Little Red Riding Hood. A transition link connects from an action in the forest episode to an action in the grandma's house episode. There does not necessarily have to be a direct dependency between the actions that are used for the *transition* link between the episodes; it is the episode dependencies that are important and that establish the transition.

There are two kinds of portals in the dynamic model: setups and conclusions. A *setup* portal is an entry point into a scenario. There may be more than one entry point into a scenario. Setups are determined by the hyperscenario storyteller as places within a hyperscenario where it is appropriate to enter the scenario. The effect of setups is to partition a scenario into meaningful chunks that can stand alone. Conversely, there may be several conclusion portals, which are exit points from a scenario. These exit points are places within the hyperscenario where it is reasonable to leave. The reasonableness of the exit point is determined by the hyperscenario storyteller.

The purpose of shifts are to move between scenarios or to outside information. A *slide* is important for branching to different scenarios. A *slide* is a shift that connects from scenario to scenario. Alternate versions of a scenario are traversed by following *slide*s. *Annotation* serve the same purpose as flashbacks, but they connect to information external to the scenario. An *Annotation* is a shift that connects to outside documentation containing background details on the scenario.

3.2.2.1 Scenario Link Operations



(a) Linear (b) Alternate Endings (c) MultiForm

(d) Flashback (e) Replay (f) Fast Forward

(g) Multiple Entries (h) Multiple Exits (i) Backstory

Figure 9: Scenario Navigation Styles

The links, portals, and shifts identified in the dynamic model for scenario navigation supports multiple styles of storytelling. Figure 9 represents nine different techniques for traversing scenarios. The vertical lines denote timelines in increasing chronological order from top-to-bottom. A *linear* style of navigation of a story is the standard chronological ordering, following a scenario from beginning to end in time sequence. This also includes following a scenario in reverse, from the end towards the beginning.

An *alternate ending* style supports the connection of different episodes to the end of a scenario. Each terminal episode has the same goal, but contains different patterns of events. For example, a “Leaving-Atlanta” scenario might have alternate episode endings that have the same goal of “Arrive-At-The-Airport”. In one episode, the sequence of events involves using a taxi, whereas the other episode involves using the subway system. The alternate ending style support navigation through the scenario where a choice is made for either ending, but the overall scenario is the same.

The dynamic model is also influenced by Murray’s concept of *multiform* stories. A *multiform* story presents a single plotline with multiple tellings [133]. These versions are the same story told from different perspectives or variants of the story with some significant detail modified. The multiform style creates parallel versions of the story that can be navigated. Each version may contain similar episodes, allowing the user of a scenario application to branch to alternate paths.

The *flashback* style supports connecting to background information within the same story. The *replay* style supports VCR-type functionality. The position within the story is adjusted back up the stream of actions, making it possible to rerun the same sequence of actions. *Fast Forward* style places markers in the scenario, letting the scenario application jump ahead in a sequence of actions.

Multiple entry scenario navigation allows a scenario application to begin the story at different places in the scenario. *Multiple exit* style is the complement of this, allowing several places to end the story. Finally, *backstory* style is similar to flashback style, except that the background information is external to the scenario. This also supports building tool-tip functionality into a scenario application, where information pops up that explains background information or gives instructions on a particular element of the scenario.

3.2.2.2 *Storytelling and Analysis Operations*

The navigational styles listed in Figure 9 can be used to construct several storytelling operations for scenarios. Each storytelling operation can be defined in terms of the links and the steps that are connected.

- ***Linear Storytelling Operation*** - A linear scenario can be represented by combining a sequence of `Setup`, `Transition`, and `Conclusion` operators on episodes to create a complete Scenario. Let `{e1,e2,e3,e4}` be the set of episodes to be contained in a `Scenario s1`. Let `{y1,y2}` be a set of episodes who elements are NOT contained in `s1`. A Linear storytelling operation using the episodes looks like:

  ```
  y1 Setup e1 Transition e2 Transition e3 Transition e4 Conclusion y2
  ```

- ***Multi-view Storytelling Operation*** - A multi-view storytelling operation involves interspersing a Linear storytelling operation with `CUT` operations to change perspectives from one `Actor` to another `Actor`. Let `{e1,e2,e3}` be the set of episodes that are contained in a `Scenario s1`. Let `y1 and y2` be episodes that are NOT contained in `s1`. Let `act1` and `act2` be `Actor` elements that perform actions that are contained within some `Episode` within `s1`[3] . In the following sequence the bracketed information represents `Episode e2` of `Scenario s1`:

  ```
  y1 Setup e1 Transition [ act1 CUT act2 ] Transition e3 Conclusion Y2
  ```

Some other storytelling operations include:

- Define entry points to a hyperscenario
- Define the exit points from a hyperscenario
- Define transitions/prerequisites between Episodes
- Define places in the scenario for jumping ahead

[3]Actions that are contained in an Event that are within an Episode.

- Define places in the scenario for replaying a sequence of actions/events/episodes
- Scenario variants - Present scenarios that have different episode patterns but meet the same scenario goal
- Episode variants - Present episodes that have different event patterns but meet the same episode goal
- Event variants - Present different patterns of actions that achieve the same event
- Action variants - Present different actions that have the same meaning in the narrative domain. Actions that have the same meaning can be interchanged in a scenario without changing the meaning of the scenario.
- Generate a successful scenario from the available scenario elements. Given goal X, what combination of episodes/events/roles/actions create a goal hierarchy that support goal X and tells a coherent story. A coherent story is meaningful and/or useful in a domain. The domain experts and the audience/participants establish what is meaningful in a domain.
- Generate an unsuccessful scenario from the available scenario elements. Given goal X, what combination of episodes/events/roles/actions creates a goal hierarchy that has one or more obstacles to achieving goal X but tells a coherent story.
- Create a shot. A *shot* is a slice of a story from a single character's point-of-view.
- Merge several shots into a single storyline.
- Storyboarding - Associate graphical images with scenario elements to create a visual representation of the story.
- Auditory - Create summary versions of scenario elements and associate them with either pre-recorded or generated audio information.
- Multimedia - Associate multimedia "stock cuts" with scenario elements to create animated/film version of the scenario.

- Translation - Process scenario elements through language translation tools to create multi-lingual scenario versions.
- Backstory - Give detailed background information on scenario elements. (Detail on actors, settings, reasons for events, etc.)
- Multiform - Use slide links to follow action(s) from one version of a scenario to another.
- Multiple Endings - Using transition links, connect to multiple final episodes (having conclusion links) in the story.
- Flashback - Use previous information in the story to help analysis at the current location in the story.
- Foreshadowing - Detect flashback links as a method to “cheat” and determine what is going to be important in the story.
- Alternate Style - Tell the story another way by using episode/event variants and following equivalence links between actions. (e.g. Replace “throw” with “pitch” and use different event patterns)

3.2.2.3 Scenario Construction and Storage

The dynamic scenario model and the static scenario elements are sufficient to construct a hyperscenario for a domain, given enough narrative information. Figure 10 is a use case scenario taken from the shuttle system description document [107]. This use case describes the successful initialization, processing, and completion of a passenger transport request. The vertical lines in the diagram (which are timelines that show the passage of time from top-to-bottom) correspond to the actors involved in the use case.

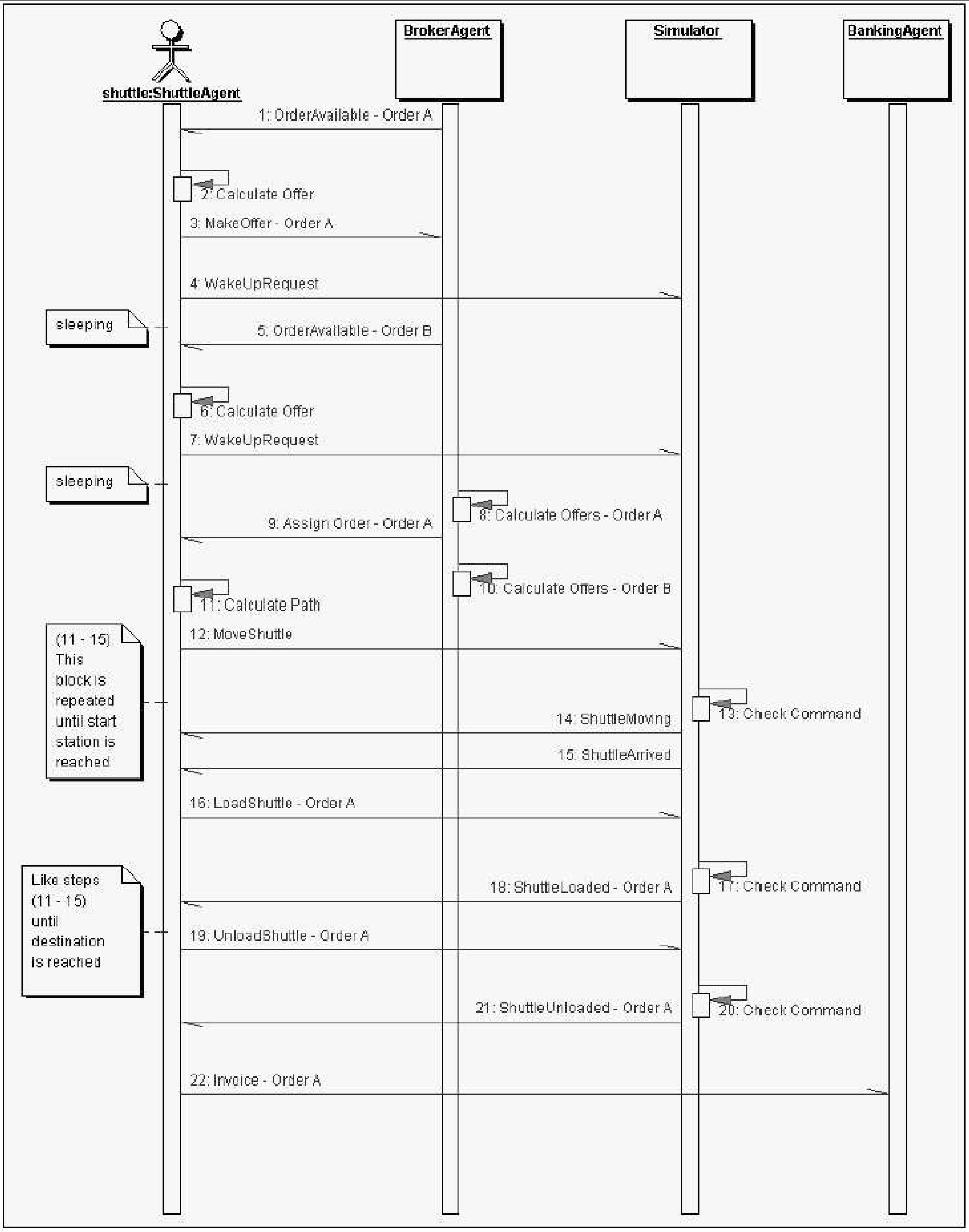

shuttle:ShuttleAgent
BrokerAgent
Simulator
BankingAgent
1: OrderAvailable - Order A
2: Calculate Offer
3: MakeOffer - Order A
4: WakeUpRequest
sleeping
5: OrderAvailable - Order B
6: Calculate Offer
7: WakeUpRequest
sleeping
8: Calculate Offers - Order A
9: Assign Order - Order A
10: Calculate Offers - Order B
11: Calculate Path
12: MoveShuttle
(11 - 15) This block is repeated until start station is reached
13: Check Command
14: ShuttleMoving
15: ShuttleArrived
16: LoadShuttle - Order A
17: Check Command
18: ShuttleLoaded - Order A
Like steps (11 - 15) until destination is reached
19: UnloadShuttle - Order A
20: Check Command
21: ShuttleUnloaded - Order A
22: Invoice - Order A

Figure 10: Shuttle Transaction Use Case

The Shuttle Agent, Broker Agent, Simulator, and Banking Agent are involved in this transaction. The messages sent between them invoke actions and constitute the actions of the scenario. The actions are numbered in sequence and follow the timelines.

Constructing a hyperscenario involves determining the events and episodes that make up the scenario, given the list the actions that occurred, in the order that they occurred. Knowledge of the shuttle domain is used to group the actions into meaningful events. The text in Figure 11 is the shuttle transaction use case formatted as a hyperscenario. The indentation in the text is used to denote the hierarchical grouping of the scenario elements. The dynamic link structures are included in the text to show potential navigation paths. To simplify the text (with the exception of the `Setup` link) only the sources of links are shown. All the actions that are included in the use case diagram appear in the hyperscenario structure, in the same sequence. The eleven identified events that group the actions are numbered in sequence within the hyperscenario. The events were determined by collecting the actions into meaningful groups. These events were then used to define the subgoals that support the scenario's primary goal. These subgoals denote the episode grouping of the events. There is only one setting for the hyperscenario, the Simulation setting, so there are no Scene changes in the hyperscenario.

The overall goal of the scenario is to `Process Passenger Request for Shuttle Transport` (`Line 1`). Each episode that is part of the scenario has goals that are subgoals that supports the primary goal of the scenario. Grouping the events that occurred by the goals that they support determines the episodes of the scenario. There are seven episodes identified in the shuttle transaction, numbered sequentially. The goal for each episode is listed on the same line as the episode.

For example, `Lines 23-38` in the hyperscenario text represent `Episode 6`, the `Transport Passenger(s)` episode. Within `Episode 6`, there are three events, `Events 7-9.` Each event contains one or more of the actions that are present in the use case diagram. `Event 8`, the `Shuttle Transport Movement` event, contains `Actions 13-15`.

Each action in the hyperscenario text lists the `Actor` that performs the action on the same line. For example, the `Simulator` Actor performs `Actions 14, 15, 17, 18, 20`, and

`21`. The links are listed in parentheses at the end of the appropriate action. There is a single entry point into the hyperscenario, the `Setup` link that is listed with `Action 1` (`Line 4`). The notation in the text shows that the `Setup` link connects from an `External` location to `Action 1`. `Action 1` also contains a `Transition` link, since it is the last action of `Episode 1` (`Lines 2-4`). The `Transition` link in `Action 1` connects to `Episode 2`.

There are six `Transition` links connecting the seven episodes going forward in sequence. There are also six `Prerequisite` links that connect the seven episodes going backwards in sequence. For example, `Action 4` contains a `Prerequisite` link that connects from `Episode 3` to `Episode 2` (`Line 11`). `Action 4` contains a link moving forward and a link moving backward, since it is the only action in the episode. There is a single exit point for the scenario, the `Conclusion` link that is in `Episode 7` on `Line 41`. The exit point is a connection from the last action of the scenario to an `External` location.

There is an example of a `Cut` link in the hyperscenario, showing the changing of perspective from one actor to another. On `Line 32` there is a `Cut` link that connects to `Action 18`. This connection switches the perspective from the `ShuttleAgent` of `Action 16` to the `Simulator` of `Action 18`. Within `Action 18`, there is another `Cut` link that switches perspective back from `Simulator` to `ShuttleAgent` by connecting to `Action 19` (`Line 34`).

```
Scenario:  Successful Shuttle Transport - Goal: Process Passenger Request for Shuttle Transport
   Episode 1: Initiate Transport Request - Goal: Handle Transport Request
      Event 1: Notify Shuttle Agent of Request
        Action 1: Send Order A Available Msg - Actor: BrokerAgent (External->Setup)(Transition->Ep2)
   Episode 2: Accept Bids For Transport Request - Goal: Accept Bid From Shuttle Agent
      Event 2: Shuttle Agent Places Bid
        Action 2: Calculate Offer - Actor: ShuttleAgent (Prerequisite->Ep1)
        Action 3: Make Offer on Order A - Actor: ShuttleAgent (Transition->Ep3)
   Episode 3: Start Shuttle Simulation - Goal: Initialize Simulated Shuttle
       Event 3: Activate Shuttle Simulation
         Action 4: Send WakeUp Request - Actor: ShuttleAgent  (Prerequisite->Ep2)(Transition->Ep4)
    Episode 4: Process Multiple Transport Requests - Goal: Process Multiple Transport Requests
       Event  4: Notify Shuttle Agent of Request
         Action 5: Send Order B Available Msg - Actor: BrokerAgent (Prerequisite->Ep3)
       Event  5: Shuttle Agent Places Bid
         Action 6: Calculate Offer - Actor: ShuttleAgent (Flashback->Ep2)
         Action 7: Send WakeUp Request - Actor: ShuttleAgent  (Transition->Ep5)
    Episode 5: Choose Shuttle for Transport - Goal: Select Shuttle to Handle Request
       Event 6: Shuttle Selection
         Action 8: Calculate Offers for Order A - Actor: BrokerAgent (Prerequisite->Ep4)
         Action 9: Assign Order to Agent - Actor: BrokerAgent
         Action 10: Calculate Offers for Order B - Actor: BrokerAgent (Transition->Ep6)
    Episode 6: Transport Passenger(s) - Goal: Transport Passenger(s) to Requested Destination
       Event 7: Determine Transport Route
         Action 11: Calculate Path - Actor: ShuttleAgent (Prerequisite->Ep5)
         Action 12: Send Move Shuttle Request- Actor: ShuttleAgent
       Event 8: Shuttle Transport Movement
         Action 13: Check Command - Actor: Simulator (Annotation->External)
         Action 14: Send Shuttle Moving Msg - Actor: Simulator
         Action 15: Send Shuttle Arrived Msg - Actor: Simulator
       Event 9: Load Passenger(s)
         Action 16: Send Load Shuttle for Order A Request - Actor: ShuttleAgent (Cut->Action 18)
         Action 17: Check Command - Actor: Simulator (Annotation->External)
         Action 18: Send Shuttle Loaded Msg - Actor: Simulator (Cut->Action 19)
       Event 10: Unload Passenger(s)
         Action 19: Send Unload Shuttle for Order A Request - Actor: ShuttleAgent
         Action 20: Check Command - Actor: Simulator (Annotation->External)
         Action 21: Send Shuttle Unloaded Msg - Actor: Simulator (Transition->Ep7)
    Episode 7: Complete Transport Request - Goal: Assign Payment for Completed Transport
       Event 11: Request Payment for Shuttle Transport
         Action 22: Send Order A Invoice - Actor: ShuttleAgent (Prerequisite->Ep6)(Conclusion->External)
```

Figure 11: Shuttle Transaction Hyperscenario

There is a `Flashback` link on `Line 16` that connects from `Action 6` back to `Episode 2`. This `Flashback` is there to associate the `Calculate Offer` action with a previous episode where the same action occurred, `Action 2` on `Line 7`.

The links that are included in the shuttle transaction hyperscenario example support navigating through the scenario using different storytelling operations. A linear navigation of this example can be done using the `Transition` links between the episodes. The `Cut` links contained on `Lines 34` and `35` support the multi-view storytelling operation as defined in section 3.2.2.2.

`Acts` are not shown in this particular example. `Acts` are presented if it is useful to have another level of grouping episodes within the scenario for a specific purpose. For example, the standard three-act dramatic structure could be applied to this example, letting `Act 1` (*Setup*) include `Episode 1`, `Act 2` (*Conflict*) include `Episode 2-6`, and `Act 3` (*Resolution*) include `Episode 7`.

The scenario conceptual model lends itself to a straightforward storage into a database system. Table 6 is a representation of how the shuttle hyperscenario can be organized into a database for accessing scenario elements. Each column denotes a database table corresponding to the scenario elements in the shuttle hyperscenario. It is possible to follow both the UML model and the database organization in the same manner to trace connections between scenario elements. For example, the database organization is used to determine that `Action 5` is an element in `Event 4` which is an element in `Episode 4` which is contained in the shuttle hyperscenario, `Scenario 1`.

Table 6: Scenario Database Organization

Scenario 1	Episode 1	Event 1	Action 1
	Episode 2	Event 2	Action 2
			Action 3
	Episode 3	Event 3	Action 4
	Episode 4	Event 4	Action 5
		Event 5	Action 6
			Action 7
	Episode 5	Event 6	Action 8
			Action 9
			Action 10
	Episode 6	Event 7	Action 11
			Action 12
		Event 8	Action 13
			Action 14
			Action 15
		Event 9	Action 16
			Action 17
			Action 18
		Event 10	Action 19
			Action 20
			Action 21
	Episode 7	Event 11	Action 22

3.2.3 Scenario Querying for Decision-Making

One of the most useful features of a structured scenario format is the ability to perform queries on stories in meaningful ways. Performing queries on hyperscenarios in a domain assumes that there is definite meaning assigned to the scenario elements and their relationships to one another. This allows queries into a hyperscenario to be based on semantic information as opposed to keyword searches for information. The primary purpose of these queries is to help make decisions about the environment being represented by the scenario. It is the audience/participant user that is interested in the querying and analysis aspects of the Hyperscenario Framework. The scenario pieces have already been identified by the domain expert, the plot has already been created by the storyteller, and the navigation structures have been inserted by the hyperscenario storyteller.

One method of representing these querying operations is a language notation called the Object Constraint Language (OCL) [183]. OCL is a notation to allow mathematical and rules based information that cannot be easily representing in UML. OCL is a constraint and a query language. A constraint is a restriction on one or more values of elements in a UML model. OCL can be used to write textual query expressions, similar to the Standard Query Language (SQL)[70] used for retrieving database information.

The following example queries are written in OCL to illustrate the kind of decision-making queries that can be performed on hyperscenario-structured information. The keyword `context` establishes which class in the model is being queried. The dot notation is used to show path expressions to sub-elements given the context. The keyword `body` delineates the start of the query operation. There are several built-in functions on sets, bags, and collections that can be used to return groups of values that are in the query result. The syntax and keywords in the following OCL constraints is taken from the OCL version 2.0 specification as referenced in [183].

- Perform scenario element lookups. Scenario lookups are useful for deciding if certain scenario elements actually occur within any of the scenarios in a system. Is there an element with a certain characteristic in the scenario? This example shows OCL code for searching for an *episode* with a specific *goal*. This type of query is useful for deciding *episode* frequency for establishing probabilities for more accurate prediction of scenarios. In the query, the `searchGoal` in the body of the query is the specific goal that is being sought. For each episode in the set of episodes contained in the scenario, the episode's goal is compared against the `searchGoal`.

```
context Scenario::findEpisode(searchGoal:Goal):Set(Episode)
body: foundEpisode -> select(Episode.Goal(searchGoal))
```

- Where are the scene changes? Can we retrieve all the scenes that occur within an episode to determine where most of the scenario occurs? This query is useful for making the decision whether or not to include scenes as part of the scenario structure. If there are no or very few scene changes, it may not be necessary or efficient to include the extra level of scenario element. The *episode* class has a query *getScenes* that will return a set of scenes.

```
context Episode::getScenes():Set(Scene)
body: scenes -> asSet()
```

- Some scenario environments may need to isolate important actions to decide a critical path for development. One way to decide on important actions in a system is to examine event changes. In the OCL query below, the *event* class has a query operation called `getTrigger` that will return a set of actions. There is an standard library function `first()` that will return the beginning of an ordered sequence. The first action in an event constitutes the trigger for the event.

```
context Event::getTrigger():Set(Action)
body: trigger -> first().Event.Action
```

- As an example of decision-making in a particular domain, the shuttle system may require the elimination of one of the shuttles. To decide which shuttle to eliminate, a query is developed to determine the number of offers for transportation requests per shuttle. The shuttle that makes the fewest offers will be the one to eliminate. Assume the shuttle domain expert has already defined a query operation on the *scenario* class called *getOffers* which will return as a set the *actions* that are the offers in the system. In the body of the query, the `madeOffers` operation will return the actions that are the offers in the shuttle system, e.g. the "bid" actions. The body of the query returns all the "bid" actions as a set called `madeOffers`.

```
context Scenario::getOffers():Set(Action)
body: episodes.events.actions.madeOffers -> asSet()
```

There are many other kinds of queries on scenarios to make decisions in a domain. For example, the actors that are serving in specific roles can be found by determining the actions that support the goal of the role. The link operations and navigational styles in the scenario can be queried to determine the variants of a scenario and the paths between scenario elements.

3.3 Scenario Grammar

The conceptual and dynamic models defined in the previous sections of the chapter are the core of the scenario ontology. Converting these models into a scenario language requires defining the syntax of the language. A *grammar* is a system of rules for describing the syntax of a language. Using a grammar, it is possible to determine if strings of information constitute syntactically valid elements of a language. The purpose of a scenario grammar developed for the Hyperscenario Framework is to give an abstract syntax for describing scenarios. This section defines the scenario grammar for the Hyperscenario Framework.

The scenario grammar is a *Context Free Grammar* (*CFG*). A CFG is a formal grammar where every production rule is of the form $V = w$, where V is a non-terminal and w is a string consisting of terminals and/or non-terminals. These grammars are context free in that any occurrence of a non-terminal such as V can be replaced with its rule w at any point.

Backus Naur Form (BNF) is a method for formal specification of CFGs [120]. The left-hand sides of BNF equations consist of non-terminals to be defined in term of other language elements. The right hand-side is a list made up of terminal symbols and non-terminals. That is, a BNF specification is a set of derivation rules, written as

```
<symbol> ::= <expression with symbols>
```

where `<symbol>` is a nonterminal, and <expression> consists of a sequence of symbols possibly separated by the vertical bar, `'|'`, indicating a choice. Any term that does not appear on the left-hand side of the equation cannot be decomposed further.

Figure 12 is the scenario grammar represented in BNF. This BNF scenario grammar was validated using the *Yet Another Compiler Compiler (YACC)* parser generator to remove any ambiguities. *Ambiguities* result when there are several ways to apply rules for a single input string. YACC takes as input a machine-readable BNF grammar and outputs a parser for the language written in the programming language C. YACC is particular suited for *Look Ahead Left to Right* (*LALR*) grammars, a subset of CFGs[44]. *LALR grammars* require that any input string can be parsed by looking ahead at most a single token. These grammars are *deterministic* in that the choice of the next rule to apply can be uniquely determined by analyzing the preceding input and a single token of the remaining input.

The tool used to analyze the scenario grammar was Bison, a YACC-compatible parser generator. The output from Bison not only includes a parser, but also an *automaton.* The *automaton* lists all the states that are possible with the grammar and the validation rules that are applied. The automaton and parser work by analyzing an input string one token at a time, deciding whether to place the token on an internal stack or to apply a rule of the grammar to reduce the current stack contents. There are two primary kinds of ambiguities in an LALR grammar that can cause conflicts: *shift/reduce* and *reduce/reduce.* A *shift/reduce* ambiguity arises when the automaton and parser have to decide whether to shift a token onto the stack or reduce the current contents of the stack. A *reduce/reduce* conflict is a more severe type of grammatical ambiguity; in this situation, the automaton can apply more than one rule for reducing the stack based on the input token.

There is one ambiguity in the scenario grammar that was not corrected. In the state where there are `actions` on the stack and the parser receives an `action` token, it has to decide whether to shift the `action` on to the stack or reduce the stack into an `event` token. The default choice of the parser generated by Bison is to shift the `action` token onto the stack. This is an acceptable solution, since every other token input (designated as the `$default` rule) causes the reduction of the stack to `event`. Appendix C is the entire automaton for the scenario grammar.

The token `verb` does not appear on the conceptual model, but had to be introduced for grammatical validity. A `verb` represents the content of an action, the actual text that describes the action. `Verb` was necessary to correct ambiguities that occur with `action` on the placement of `actor` and `prop` elements.

```
%token goal prop actor setting role
%token intermission manifest verb
%%
scenario : goal cast inventory episodes
         | goal cast inventory acts ;
cast : /* empty */ | cast character ;
character : prop | role | actor ;
inventory : /* empty */ | manifest props ;
props : prop | props prop;
acts : act | acts act ;
act : episodes intermission ;
episodes : episode | episodes episode ;
episode : goal events | goal scenes ;
scenes  : scene | scenes scene ;
scene : setting events ;
events : event | events event ;
event : actions ;
actions : action | actions action ;
action : verb | verb prop | actor verb | actor verb prop ;
%%
```

Figure 12: Scenario Grammar in BNF notation

At the top of the grammar listing in Figure 12 are two lines listing the tokens for the grammar. These tokens are the terminal symbols of the language. As terminal symbols, how these tokens are actually represented in an input string for a language are domain specific. For example, a noun looks different in English than in Russian, but the `noun` token is used in the grammars of both languages.

With the exception of `verb`, `intermission` and `manifest`, each token corresponds directly to an object in the scenario conceptual model. The extra tokens are place holders to designate the beginning or ending of the non-terminal symbols `act`, `action`, and `inventory`. These place holders are necessary grammatically and serve the same purpose as punctuation characters do for programming languages. For example, without the `intermission` token, it is ambiguous when (given an `act` token) to reduce the stack into `acts` or into a completed `scenario`.

The scenario storyteller can use the grammar to build tools for parsing, generating, or analyzing scenario strings. The availability of the scenario grammar makes it possibles to determine the validity of scenario element strings. For example, let us say that we are given strings of scenario elements for the shuttle system with the problem of deciding whether or not the strings are valid scenarios. The symbols from the scenario grammar are used in the strings to make the analysis straightforward.

```
input1 = goal goal action action action goal action action goal action action action
input2 = goal character character manifest prop goal action action goal action action action
input3 = goal goal character character action action action goal action action action
```

Each input string is parsed using the scenario grammar to determine if the appropriate reductions will produce a scenario. The results of parsing all three strings are:

```
input1 = goal episodes (valid scenario)
input2 = goal cast inventory episodes (valid scenario)
input3 = ???  (not valid)
```

Input string $\texttt{input}_3$ cannot be parsed and reduced into a pattern of symbols that represent a valid scenario. Although the grammar will allow us to decide that $\texttt{input}_1$ and $\texttt{input}_2$ are syntactically valid scenarios, it is still possible for them to be semantically incorrect. The meaning of the strings are not reflected in the grammar, just their syntaxes.

3.4 Scenario Language Implementation

The language implementation of the scenario grammar has to support two objectives: 1) representation in a form that is general enough to describe the structure of any scenario and 2) expressive enough to support different views of the data. The choice of a markup language supports both these objectives. *Markup languages* are languages that describe a document's structure using embedded tags, leaving presentation details to the capabilities of the structure-aware applications. Alternatively, a *text formatting approach* describes the appearance of the document by embedding special commands or control sequences, such as font style or margin settings. Most conventional word processing software applications use a text formatting approach. The problem with text formatters is that the encodings are proprietary and platform-dependent. Each software manufacturer may describe a proprietary text formatting technique.

For many documents, the important aspect is not how the document looks but how it is put together. Constructing documents using a standard structural notation would allow the presentation of the documents to be handled by the tools that displays them. This was the reason for the development of the *Standardized Generalized Markup Language* (*SGML*) specification in the mid-1980s. It originated from the *Generalized Markup Language* (*GML*) invented by IBM in 1969 [139]. SGML was developed by the *International Standards Organization* (*ISO*) to describe the structure and content of electronic documents. Instead of embedding non-printable key sequences to control format, specialized tags are defined to represent document structure. For example, the beginning of a paragraph is denoted using a tag of the form `<paragraph>` follow by the paragraph's content. The end of the paragraph is then denoted using a corresponding end tag, in this case `</paragraph>`. By defining the appropriate structural tags, complex documents are represented separately from their formatting.

The SGML standard was quickly adopted by the U.S. government as a method of generating platform independent documentation for equipment and weapons systems. The *Continuous Acquisition and Life-Cycle Support* (*CALS*) initiative from the middle 90's relied heavily on SGML as an interchange format for documents [28]. However, SGML proved rather difficult and complex to use. SGML is a meta-language, a language to describe other languages. Each subset of SGML uses its core concepts, but defines tags specific to a particular domain. The use of SGML or SGML-based languages was limited outside of government applications because of its complexity and size. Some word processing packages, such as WordPerfect and FrameMaker, used SGML to create ASCII versions of their documents.

The use of markup languages changed significantly with the introduction of *HyperText Markup Language* (*HTML*) in 1989. HTML is a markup language for web documents that describe the structure while allowing a web browser to handle the document presentation [69]. HTML is now the most widely used SGML-based language. The structural flexibility and the extensibility of HTML is limited because it was intended to describe web pages handling different types of content. HTML is an attempt to express hypertext in a computer-readable document. The term *hypertext* is defined as:

> A special type of database system, invented by Ted Nelson in the 1960s, in which objects (text, pictures, music, programs, and so on) can be creatively linked to each other [127].

The concept of hypertext was predicted in a seminal article by linguistic scholar Vannevar Bush [19]. The hypertext community considers the word *hypertext* in HTML a misnomer because HTML is lacking many of the characteristics it considers vital for hypertext. For example, the idea of highlighting a link to make it stand out from the rest of the text is not a hypertext convention. Hypertext was intended to be more meaningful than a collection of links from one location in a document to a location in another document. HTML linking has made the Web as widespread as it is today but there are several problems that limit its flexibility:

1. *Lack of control of information retrieval* - When a link is selected in the conventional HTML document, there is no control on how much of information will be displayed. An entire Web document is retrieved and inserted into browser window, replacing the previous content. The most that can be done to control the information is positioning it in the browser window.

2. *Unknown target type* - The data type of the destination document is also unknown before traversing the link. The name of the link may give some information as to what is being pointed to, but it is not always accurate.

3. *One-way linkages* - HTML hyperlinks are always one way. There is no direct association between the originating document and the target document to the link; there is no path between them. Using the browser's "back" button causes navigation using the history file, not an explicit path.

4. *Dangling references* - A dangling reference occurs when a link is selected and returns the infamous *404 File Not Found* message. This means that the data has moved, but the link hasn't been updated, or the the link has been entered incorrectly.

5. *Fixed anchoring* - A fixed anchor refers to a link that is anchored in a specific location within a document.

The generic nature of HTML and these linking problem make it content neutral and not nearly expressive enough to represent the scenario model.

XML is a meta-language that was developed with the best features of SGML [188]. It allows for the creation of domain-specific languages. It also corrects many of the linking, structural, and semantic problems of HTML. XML was chosen as the implementation mechanism for the scenario language.

3.4.1 XML Overview

XML is a major subset of SGML. Much of the complexity and many of the seldom used features within SGML were removed to create XML. With XML, it is easier to describe domain specific documentation structures. Not only is the structure of a document apparent in XML markup, but its semantics are available as well. This means that applications can manipulate XML documents without having to hard-code information about the document's application. The structure of the document gives its meaning.

A developer creates a grammar for an instance of XML for a domain. This can done by defining a *Document Type Definition* (*DTD*). A DTD is the grammar of an XML language. It is a list of the elements, entities, notations, and attributes that are available to the language. It establishes the relationships between these elements and how they can be nested within one another.

It is not necessary for every XML document to contain an embedded DTD or have a reference to an external DTD. The XML specification defines two types of documents: *valid* and *well-formed*. A *valid* document is one that adheres to a specific DTD. Application tools can use the DTD to validate the structure of XML document in that particular language. A *well-formed* document does not require a DTD. This kind of document is considered acceptable as long as it follows standard rules for creating tags and structuring an XML document.

Linking in XML documents is done using *XML Linking Language* (XLL) [35]. XLL is a far more robust linking capability than that found within HTML. In XLL, a link is considered a relationship between resources. This relationship depends on the application that is processing the link and the semantics supplied with the link. The XLL standard supports several new capabilities, including bidirectional linking, persistent links, dynamic updates, support for annotations, and link management. Link types can be created in XLL through named links. These named links support traversal behavior based on the links' types. There are two components of XLL that make this possible: *X-Link* and *X-Pointer*. X-Link introduces a division of links into *simple* and *extended*. *Simple* links are very similar to those currently in HTML and are usually in-line (embedded as part of an anchor tag). The *extended* links are used to express relationships between more than two resources. These links can be out-of-line, meaning that they are contained in external files or link databases. This makes link management more straightforward and leverages the use of databases for configuration management. The *X-Pointer* definitions are an advanced addressing scheme for XML documents. X-Pointer is a notation for pointing to the sub-parts of a document without requiring an identification number or name within an anchor tag. X-Pointer is a syntax for accessing resources by traversing an element tree of the document. The element tree is based on the *Document Object Model* (*DOM*) standard. The DOM is a platform- and language-neutral interface that allows program and scripts to dynamically access and update the content, structure, and style of documents [187]. The DOM provides a standard set of objects for representing XML documents, a model for how the objects can be combined, and an interface for accessing them. This greatly simplifies the construction of tools for XML documents. For example, the X-Pointer reference:

```
child(2,episode)(3,.)
```

uses DOM syntax to locate the third child element of the second episode in the scenario document.

3.4.2 Scenario Markup Language (SCML)

The XML syntax and the scenario grammar have been implemented into the *Scenario Markup Language* (*SCML*). SCML tags correspond to the concepts and rules from the scenario ontology. Rules for combining and nesting tag elements are derived from the associations and multiplicities represented in the conceptual schema.

The BNF representation of the scenario grammar defined in the previous section can be ported directly into XML DTD notation. The XML DTD notation is similar to BNF (consisting of terminals, non-terminals, symbols, and production rules) along with the following additional constructs:

- angle braces ”< >” enclosing definitions
- suffix "*" for Kleene closure (a sequence of zero or more of an item)
- suffix "+" for one or more of an item
- suffix "?" for zero or one of an item
- vertical bar ”|” separating alternatives
- parenthesis ”()” for grouping items

Tags that represent objects in an XML language are defined using the `ELEMENT` keyword and are equivalent to non-terminals in an BNF grammar. For example, we can define an address book element called a `person`. As an element in a DTD, the declaration looks like this:

```
<!ELEMENT person (name, e-mail*) >
```

In this example, the `ELEMENT person` is made up of other elements: a `name` and optionally `e-mail` address elements. The declaration for each of these sub elements are contained within the DTD. Each element may have attributes associated with it. These attributes establish the properties that can be modified for the element. External documents can be referenced using the keyword `ENTITY` within DTDs. Specific symbols or constants can be declared using the keyword `NOTATION`. Keywords that represent the valid XML Datatypes that can be used on elements and their attributes are listed in Table 7.

Table 7: XML Datatypes

Keyword	Description
CDATA	Character Data
ENTITY	An entity declared in an XML DTD
ID	A unique element identifier
IDREF	The value of a unique ID type attribute. An IDREF references the ID of another XML element
NMTOKEN	An XML name token that consists of letters, digits, underscores, hyphens, or periods
NOTATION	A notation declared in an XML DTD

Figure 13 is the definition and attribute list for the SCML `hyperscenario` element. The `hyperscenario` element is the root element of the SCML DTD. The attribute list includes the minimum characteristics necessary for a `hyperscenario.` The only attribute that is necessary is the `title` of the scenario. The keyword `#REQUIRED` states that the attribute must appear with the element`.` The `goal` of the scenario is the first element contained within the `hyperscenario` tag. An identification number, `hyperscenarioID`, is an attribute that may be present but is not required for a `hyperscenario` tag. The keyword `#IMPLIED` means that the attribute does not have to appear. Many of the SCML element attributes are `CDATA`. `CDATA` are alphanumeric strings of characters. The `settings` attribute contains a listing of the possible locations within the scenario.

```
<!ELEMENT hyperscenario (goal, cast?, inventory?, (episode+|act+))>
<!ATTLIST hyperscenario
     hyperscenarioID ID #IMPLIED
     title NMTOKEN #REQUIRED
     settings CDATA #IMPLIED>
```

Figure 13: SCML Root Element Definition and Attributes

Figure 14 displays the SCML `cast` and `character` ELEMENT definitions. The `cast` ELEMENT contains a series of one or more `character` ELEMENTs . The `inventory` ELEMENT contains one or more `prop` ELEMENTs. A character ELEMENT contains either a `role` ELEMENT or an `actor` ELEMENT or a `prop` ELEMENT. Note that the `role` ELEMENT contains one or more `goal` ELEMENTs within it.

```
<!ELEMENT cast (character+)>
<!ELEMENT inventory (prop+)>
<!ELEMENT character (role|actor|prop)>
<!ELEMENT role (goal+)>
<!ATTLIST role
     roleID ID #REQUIRED
     name NMTOKEN #REQUIRED>
```

Figure 14: Cast and Character Element Definitions

The `actor` ELEMENT contains a unique identification number, called `actionID`, and a required `name` attribute (Figure 15). The `prop` element has an attribute `targetOF` that can be set to the value of the `actionID` of an `action` ELEMENT that manipulates it.

```
<!ELEMENT actor (#PCDATA)>
<!ATTLIST actor
     actorID ID #REQUIRED
     name NMTOKEN #REQUIRED>

<!ELEMENT prop (#PCDATA)>
<!ATTLIST prop
     propID ID #REQUIRED
     name NMTOKEN #REQUIRED
     targetOF IDREF #IMPLIED>
```

Figure 15: Actor and Prop Element Definitions

The `act` ELEMENT is a grouping of `episode` ELEMENTs. It is shown in Figure 16.

```
<!ELEMENT act (episode+)>
<!ATTLIST act
     name NMTOKEN #IMPLIED>
```

Figure 16: Act Element Definition

An `episode` ELEMENT always contains a `goal` ELEMENT, as shown in Figure 17. It then contains either a grouping of `scene` ELEMENTs or a grouping of `event` ELEMENTs.

```
<!ELEMENT episode (goal, (scene+|event+))>
<!ATTLIST episode
     episodeID ID #IMPLIED
     name NMTOKEN #IMPLIED>
```

Figure 17: Episode Element Definition

Figure 18 is the `goal` ELEMENT definition and its corresponding attribute list. Having goals as SCML elements supports focused searches based on the goals of `characters`, `events`, `episodes`, and entire `scenarios`.

```
<!ELEMENT goal (#PCDATA)>
<!ATTLIST goal
     goalID ID #IMPLIED
     name NMTOKEN #IMPLIED>
```

Figure 18: Goal Element Definition

The `scene` and `setting` ELEMENTs appear in Figure 19. The `setting` element that is included within the `scene` is the name of the location of the `scene`.

```
<!ELEMENT scene (setting, event+)>
<ATTLIST scene
    sceneID ID #IMPLIED
    name NMTOKEN #IMPLIED>

<!ELEMENT setting (#PCDATA)>
```

Figure 19: Scene and Setting Element Definitions

The `event` and `action` ELEMENTs appear in Figure 20 on page 74. Both elements may or may not have `goal` ELEMENTs associated with them. This supports random events and actions that happen during a story that do not serve a particular purpose or reason in the story. An `event` ELEMENT must have at least one `action` ELEMENT to be considered an `event`. The `actor` that performs that action must appear within the ELEMENT. A `prop` may be the target of an `action`, so it is a sub-element in the production rule for the `action`.

The `storylink` element defines all the link structures that can be used to navigate through the scenario. There may be multiple links present within an action, generating many alternate pathways through a story. The `target` attribute contains the name of the object being manipulated. This makes it possible to determine all actions that interact with a particular `prop` in the scenario by a search based on `action` targets.

```
<!ELEMENT event (goal?, action+)>
<!ATTLIST event
     eventID ID #IMPLIED
     name NMTOKEN #IMPLIED>

<!ELEMENT action (goal?, actor, prop?, storylink*)>
<!ATTLIST action
     actionID ID #IMPLIED
     name NMTOKEN #IMPLIED
     originator IDREF #IMPLIED
     target IDREF #IMPLIED>
```

Figure 20: Event and Action Element Definition

Figure 21 is the SCML element definition for a `storylink`. The semantics of these links type is decided by the particular application. The definition for `storylink` lists the different types of links that are possible. Every link has to have a `target` of the link as an attribute. Classes or categories of links are entered as values in the `class` attribute. The `xml:link` attribute is the method for including XLL functionality in the language. Recall that XLL links can be either `simple` or `extended`; here the default is `simple`. An `extended` link allows multiple selections and targeting to aliased locations. The `show` attribute is a very powerful feature of XLL. Instead of the standard replacement of the resource being viewed, a link can be set to embed the retrieved information in the current document or spawn a new window for display of the information. The `actuate` characteristics determines if the link is traversed by direct manual selection of the user or is automatically followed based on some implicit behavior. For example, a link may be initiated by the movement of the mouse over a certain area of the display.

```
<!ELEMENT storylink (annotation|flashback|transition|
     cut|slide|dependency|equivalence|setup|conclusion)>
<!ATTLIST storylink
     source ID #IMPLIED
     target IDREF #REQUIRED
     class CDATA #IMPLIED
     xml:link CDATA #FIXED "simple"
     href CDATA #IMPLIED
     role CDATA "transition"
     title CDATA #IMPLIED
     show (embed|replace|new) "new"
     actuate (auto|user) "user"
     behavior CDATA #IMPLIED>
```

Figure 21: Storylink Element Definition

The individual link element definitions (Figure 22) are empty tags. The attributes of the `storylink` tag that enclosing a link determine its characteristics.

```
<!ELEMENT flashback EMPTY>

<!ELEMENT cut EMPTY>

<!ELEMENT annotation EMPTY>

<!ELEMENT slide EMPTY>

<!ELEMENT dependency EMPTY>

<!ELEMENT equivalence EMPTY>

<!ELEMENT prerequisite EMPTY>

<!ELEMENT rewind EMPTY>

<!ELEMENT setup EMPTY>

<!ELEMENT conclusion EMPTY>
```

Figure 22: SCML Link Definitions

The complete SCML DTD appears in Appendix A. Figure 23 is an excerpt (the first three episodes) of the shuttle scenario from Section 3.2.2.3 written in SCML. The entire text of this shuttle scenario in SCML appears in Appendix E.

```
<hyperscenario title="Successful Shuttle Transport">
  <goal>Process Passenger Request for Shuttle Transport</goal>
  <episode episodeID=1 name="Initiate Transport Request">
    <goal>Handle Transport Request</goal>
    <event eventID=1 name="Notify Shuttle Agent of Request">
      <action actionID=1 name="action1">
        <actor>BrokerAgent</actor>
      Send Order A Available Msg
        <storylink source="."><setup/></storylink>
        <storylink target="#action2"><transition/></storylink>
      </action>
    </event>
  </episode>
  <episode episodeID=2 name="Accept Bids For Transport Request">
    <goal>Accept Bid From Shuttle Agent</goal>
    <event eventID=2 name="Shuttle Agent Places Bid">
      <action actionID=2 name="action2">
        <actor>ShuttleAgent</actor>
      Calculate Offer
        <storylink target="#action1"><prerequisite/></storylink>
      </action>
      <action actionID=3 name="action3">
        <actor>ShuttleAgent</actor>
      Make Offer on Order A
        <storylink target="#action4"><transition/></storylink>
      </action>
    </event>
  </episode>
  <episode episode=3 name="Start Shuttle Simulation">
    <goal>Initialize Simulated Shuttle</goal>
    <event eventID=3 name="Activate Shuttle Simulation">
      <action actionID=4 name="action4">
        <actor>ShuttleAgent</actor>
      Send WakeUp Request
        <storylink target="#action3"><prerequisite/></storylink>
        <storylink target="#action5"><transition/></storylink>
        </action>
    </event>
  </episode>
```

Figure 23: Shuttle Scenario in SCML

3.5 Creating Scenarios in SCML

There are several techniques for finding stories in a domain, but it is necessary first to describe features that a narrative domain needs to effectively use SCML. There are also limitations to the use of SCML, based on both the implementation choice and the scope of potential scenario applications.

3.5.1 Requirements for using SCML for a Narrative Domain

SCML was developed from a narrative morphology that sought to incorporate story elements that were common across domains. The following is a list of characteristics of a domain, not necessarily organized by priority, that enables SCML encoding.

1. **A goal network is required.** A narrative domain has to have an explicit goal network or one that can be derived through enterprise or domain modeling. These goals and sub-goals can be associated with episodes, events, actions, and roles in a straightforward manner.

2. **Subject Matter Experts (SMEs) have to define canonical stories**. To be useful for decision-making and worth the trouble of SCML encoding, there needs to be a set of recurring stories or themes. Knowledge representation using narrative and SCML is an approach to capturing expert system knowledge–the problems and standard solutions within a domain. The computer-readable form of SCML, along with the appropriate rule-base make it possible to generate and analyze unexpected stories, but there has to be a base of standard occurrences.

3. **The domain must have definitive action/actor pairs.** There must be task lists, activities, or commands that can be associated with particular subsystems, roles, or characters. There is a definition of a screenplay that describes it as "character in action"[36]; the minimum for building a story is known characters accomplishing known behaviors. SCML encoding requires at the very least the *who* doing *what* of a story; the *when*, *why*, *how* and other semantics of the actions are helpful to add context.

4. **Domain ontology has to be mapped to SCML framework.** The reason for defining the scenario ontology and then implementing the SCML language is that ontologies are at the appropriate level of semantic interchange. To be effective with SCML, a domain's ontology has to be connected to the scenario ontology. There are two main approaches to doing this mapping: Information flow and use of existing ontology languages.

   (a) **Information-Flow Based Ontology Mapping.** In [101], Kalfoglou and Schorlemmer proposed a technique for ontology mapping that was grounded in information flow and channel theory. Their approach presupposes the flow of information in an ontology and uses the mathematical model found in channel theory to describe this flow. Channel theory uses *local logics,* which are types, instances, and binary relations that describe the vocabularies of a community. By characterizing the local logics as ontologies, the researchers were able to develop a formal technique for accomplishing the mapping between two ontologies. Using this method, the narrative domain to be mapped to the scenario ontology needs to be described as a ontological tuple that include the concepts, symbols, and relations on types and instances.

   (b) **Mapping to existing ontology language**. Ontology languages represent a domain in terms of concepts, classes, and slots. There are several languages used to describe a narrative domain, including Ontolingua [51], RDF [171], and DAML+OIL[123].

3.5.2 Limitations of SCML and the hyperscenario framework

There are limitations in the scope of use of SCML and potential applications that manipulate SCML-encoded artifacts. When the development of the hyperscenario framework first began, DTD representations for XML languages were the most mature of the techniques. There was also a reasonable availability of tools to parse and exchange DTD-based XML documents. Since then, other approaches, most notably *XML Schema* (*XML-S*), have corrected some of the problems inherent in DTDs. Tools for these other XML technologies are now more widely available.

Some of the limitations are due to DTD or language syntax constraints. There is also some difficulty in trying to capture all variations of narrative in one general language. One of the most useful characteristics of SCML as a markup language is its flexibility for handling domain specific objects that are not part of the base language definition. As an XML language, SCML was designed to be extended when necessary to capture domain specific story elements. The use of namespaces, notations, entities, and modifications of the SCML DTD allow it to be extended within a particular domain. The following is a list of some of the disadvantages in SCML and the hyperscenario framework.

1. **No Ternary Relationships in the Conceptual Model**. All of the associations between classes in the UML model of the scenario ontology are between two elements at a time. The presence of ternary (three-way) relationships more tightly couples elements to support certain kinds of queries. For example, if there was a ternary relationship in the conceptual model between `Actor, Action,` and `Prop` it becomes possible to determine what `Actor` manipulated a `Prop` with a single query.

2. **The SCML DTD format is weakly typed.** DTDs were the basis for creating XML-based languages when the technology was first introduced. However, DTDs suffer from several disadvantages such as the lack of user defined types. The loose typing of DTD-based languages make it difficult to validate the form of the data in a document. This weak typing is why SCML entity declarations are used to define enumerated types.

3. **DTDs are not XML compatible**. The DTD format is not written in XML. This makes it necessary to have external tools for validation. It also fails to leverage the vast number of XML tools available for parsing and analysis.

4. **No language structure for handling ambiguous events.** Ambiguous event patterns cannot be handled in SCML. If the pattern of actions resolve into more than one event, the application has to determine which event might have occurred. Associating

probabilities and weighting factors to disambiguate between similar events is necessary to create reasonably accurate stories. This is accomplished by inserting calls to list processing languages, like LISP or Prolog, within the code, or merging SCML with production rule systems like ACT-R or SOAR [5].

5. **No axioms, rules embedded directly in language structure.** There are syntactic rules that are part of the language structure, but no method in a DTD-based language for including rules on multiplicity, transitivity, and other properties that influence how scenario elements can be combined

3.5.3 Heuristics for finding scenarios for SCML Encoding

Analyzing the artifacts and processes of the organization helps determine episodic information to create reasonable stories. Their is a wealth of existing literature on finding stories, at the enterprise modeling level ([148] and [93]), domain modeling [96], and in the hypertext community [156]. Techniques that attempt to capture the goals, motivations, characters involved, or tasks inherent in a system can also be used to model stories. The technique chosen for story discovery depends upon the domain, but there are some general guidelines for finding stories and increasing the amount of coverage of information. One such set of guidelines is the contextual inquiry technique developed by Holtzblatt and Beyer to aid requirements analysis and system design [89]. *Contextual inquiry* is the basis of customer-centered design, an attempt to involve the customer in all aspects of software development. To find out about customers' requirements, the inquiry technique is used to construct work models that describe their environment. These models are used to highlight potential design problems based on the current system. Although contextual inquiry was developed as a technique for software design, it is very useful in capturing stories about the users' domain. Holtzblatt and Beyer present general guidelines for five work models defined in the technique.

- **Flow Model** – The flow model represents the communication within a system with respect to work flow. This model is used to define how work is broken up across an organization and how collaboration is accomplished to get work done. Within the model, the designer determines the individuals involved, their responsibilities, the groups that coordinate work products, and the artifacts that represent the work product. This model offers a broad view of the organization, showing the people and their responsibilities. Determining the force structure to support a military mission objective is an example application of flow modeling.

- **Sequence Model** – A sequence model is a list of work tasks and their appropriate ordering. The focus of a sequence model is discovering a customer's intent, based on the actions done. Individual tasks are decomposed into the individual actions and analyzed. The order, triggers of actions, potential breakdown in task completions, even hesitations and errors are determined in this model.

- **Artifact Model**– An artifact model is a description of the work products that is manipulated by the system. Work products are created, used, and modified in the course of doing work. These artifacts and their manipulation give insight into the work practices and business rules of an organization. This model is looking for the information presented by an object, its structure, and the presentation of the object, how it is used and any conceptual distinctions represented in the object.

- **Cultural Model** – The cultural model describes organizational responsibilities and expectations, the influence of the work environment on accomplishing tasks. Work does not happen in a vacuum; it is influenced by the culture that defines the expectations, desires, policies, and value system of the work. The cultural context drives how things are done based on standards, policies, and organizational influences.

- **Physical Model** – A physical model describes the physical environment where work occurs. For example, the inside of a tank, the top of a bridge, or a conventional office. This model reveals the design constraints for new system development. The model places a distinction on describing the site, the workplace, hardware, software, communication lines, and layout, including even the organization of materials. There can be many organizational assumptions derived from how space is utilized.

Although contextual design uses certain diagramming conventions, it is not prescriptive. The purpose of the technique is to have a systematic approach for modeling the design of a system.

3.6 Summary

This chapter describes the Hyperscenario Framework derived from the literature review and the study of narrative morphology of Chapter two. An automated shuttle system example was introduced at the beginning of the chapter to motivate the framework and serve as a running example.

Table 8 lists the models and representations in the Hyperscenario Framework. The first column is the concepts identified in the conceptual model. Column two lists the linking operations identified in the dynamic model for scenario navigation and storytelling. The first two columns are the basis for the scenario ontology. An ontology is a description of a domain in terms of its vocabulary, concepts, relationships, rules, and properties. The third column is the symbols of the abstract syntax grammar developed from the conceptual model. As a context-free grammar, the scenario grammar of the Hyperscenario Framework is defined with a set of production rules in BNF notation. The language as depicted in the fourth column merges all the concepts, link operations, and grammar elements into a scenario specification.

Table 8: Hyperscenario Framework

Conceptual	**Dynamic**	**Grammar**	**Language**
Scenario		Scenario	Hyperscenario
Cast		Cast	Cast
Inventory		Inventory	Inventory
Character		Character	Character
Role		Role	Role
Actor		Actor	Actor
Prop		Prop	Prop
Goal		Goal	Goal
Act		Act	Act
Episode		Episode	Episode
Scene		Scene	Scene
Setting		Setting	Setting
Event		Event	Event
Action		Action	Action
	Link		Storylink
	Equivalence		Equivalence
	Dependency		Dependency
	Rewind		Rewind
	Setup		Setup
	Transition		Transition
	Prerequisite		Prerequisite
	Flashback		Flashback
	Conclusion		Conclusion
	Slide		Slide
	Annotation		Annotation
		Intermission	
		Manifest	
		Verb	

XML was chosen as the method for representing the scenario language because of the characteristics of markup languages. Markup languages are a powerful technique for representing the structure of artifacts, separating them from the rendering and formatting applications. The scenario language of the Hyperscenario Framework, called Scenario Markup Language (SCML) incorporates the concepts and features from the scenario ontology and grammar. There are some limitations on SCML, much of it based on its DTD syntax. The final section in this chapter also includes some general heuristics for finding stories in a narrative domain for representation in SCML.

Chapter 4

SCENARIO APPLICATIONS

Knowledge is of no value unless you put it into practice.

– Heber J. Grant

This chapter describes software applications and narrative domains illustrating the hyperscenario framework. This research successfully separated the scenario structure from its use; the following sections outline the affordances created by this structure. The first section describes a scenario generator application and its actual implementation as a proof-of-concept. We developed a prototype of the scenario generator, called *Aesop*, to show how low-level actions and domain knowledge are applied to create complex, canonical stories. The second section discusses military applications for SCML, particularly with modeling and simulation environments. The semantic information for battle planning, simulation interchange, and decision support is captured in scenario artifacts. The final section outlines a potential use of SCML for software design patterns.

4.1 Scenario Generation

Each narrative domain that uses the hyperscenario framework would require tools for parsing and manipulating SCML-based narratives. The rules and policies of a domain assign semantics to SCML story elements. Applications such as scenario walk-throughs, scenario generation, and scenario manipulation have component pieces in common. We chose a scenario generator as an example application to show some of the capabilities of SCML.

4.1.1 Scenario Generator Architecture

Figure 24 on page 88 depicts the architecture of an automated scenario generator. This application is designed to generate a story from low-level event data. The event trace consists of actions/steps gathered from a log file, e-mail messages, or output from a simulation. The main impact of this scenario generation approach is the ability to add "plot structure" to an event trace. Instead of a list of actions and steps in a linear sequence, the encoded narrative can contain contextual information about the environment under which the actions took place. For the purposes of discussion, the format of the event trace is a log file generated by an external application.

Each component in Figure 24 is responsible for adding information to the plot structure. These components could be software systems, direct user input, or a combination of automated and human intervention. The purpose of the *Filter* component is to remove non-relevant background information. This can be header information or tool-related commands that add no context to the story. The *Parser* examines the log file, building action tokens. An *action token* is a series of fields, such as action name, type, actor, setting, and time-stamp that could be gleaned from each transaction (See Figure 47 on page 133). It is not necessary to have every field to construct a story; the minimum is a list of actions/actor pairs. The *Semantic Analysis* is the first of the components that would have knowledge of the narrative environment. The Semantic Analysis component searches for event patterns known to occur in the domain. It has a list of significant events and the collections of actions that could make up each one of them. There is also information on actions that cause event transitions and inter-action dependencies in this component. Semantic analysis generates event pragmas (fragments) from the action tokens. The majority of the story creation work happens in the *Inference Engine.* Episodes can be determined by doing pattern matching on the sequence of events. If the engine detects an event pattern that matches, it can assume the goal for that series of events and construct an episode. A similar style of pattern matching is used to detect the specific scenario that occurred. Once the scenario structure is complete, the inference engine can add other narrative information, such as cast of characters and scene changes. Finally, the inference engine uses the appropriate style sheet to generate the encoded narrative.

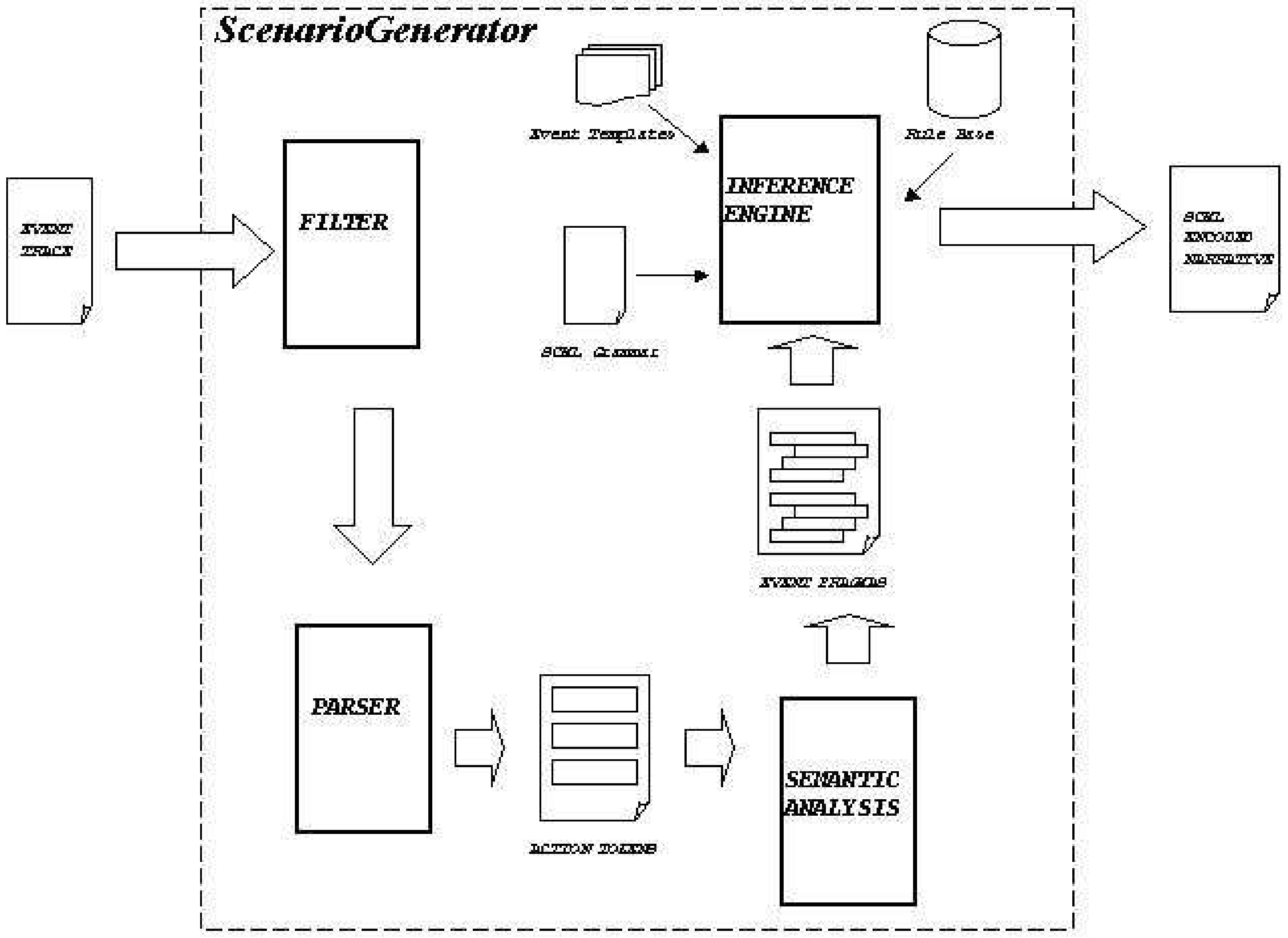


Figure 24: Scenario Generator Architecture

4.1.2 Scenario Generator Prototype

A proof-of-concept scenario generation tool has been developed for SCML. To serve as a proof-of-concept, an environment needed to be chosen that had: 1) a narrative basis, 2) differing levels of detail, and 3) support for multiple perspectives. It also had to be sufficiently self-contained to describe a manageable number of actions, events, and episodes for analysis. The choice of a three-dimensional maze game simulation supports these characteristics. The storytelling (or story experiencing) aspect of the maze that makes it an appropriate choice for the prototype. Movement through the maze and accomplishing tasks/sub-goals can be described from the standpoint of narrative. The implemented prototype tool is called *Aesop*, after the legendary Greek fabulist [34], to highlight its storytelling focus.

Aesop consists of Java applets and *Common Gateway Interface (CGI)* Perl scripts written as a simple three-dimensional maze game. The tool is written as applets to leverage existing Internet browsers for the front-end *Graphical User Interface (GUI)* of the story generator. Figure 25 shows the design for Aesop. The applets captures the user's interaction and transmit information to the Perl scripts to supply the context. The plot that is created by this context is then encoded in SCML. Transformations are applied to the SCML-encoded document to create alternate representations of the same story. External program code, written in *XML Style Language (XSL) [128]*, parses the document for rendering and transformations. Alternate story views such as hierarchical, summary, tabular, and literary narrative are made available for the player.

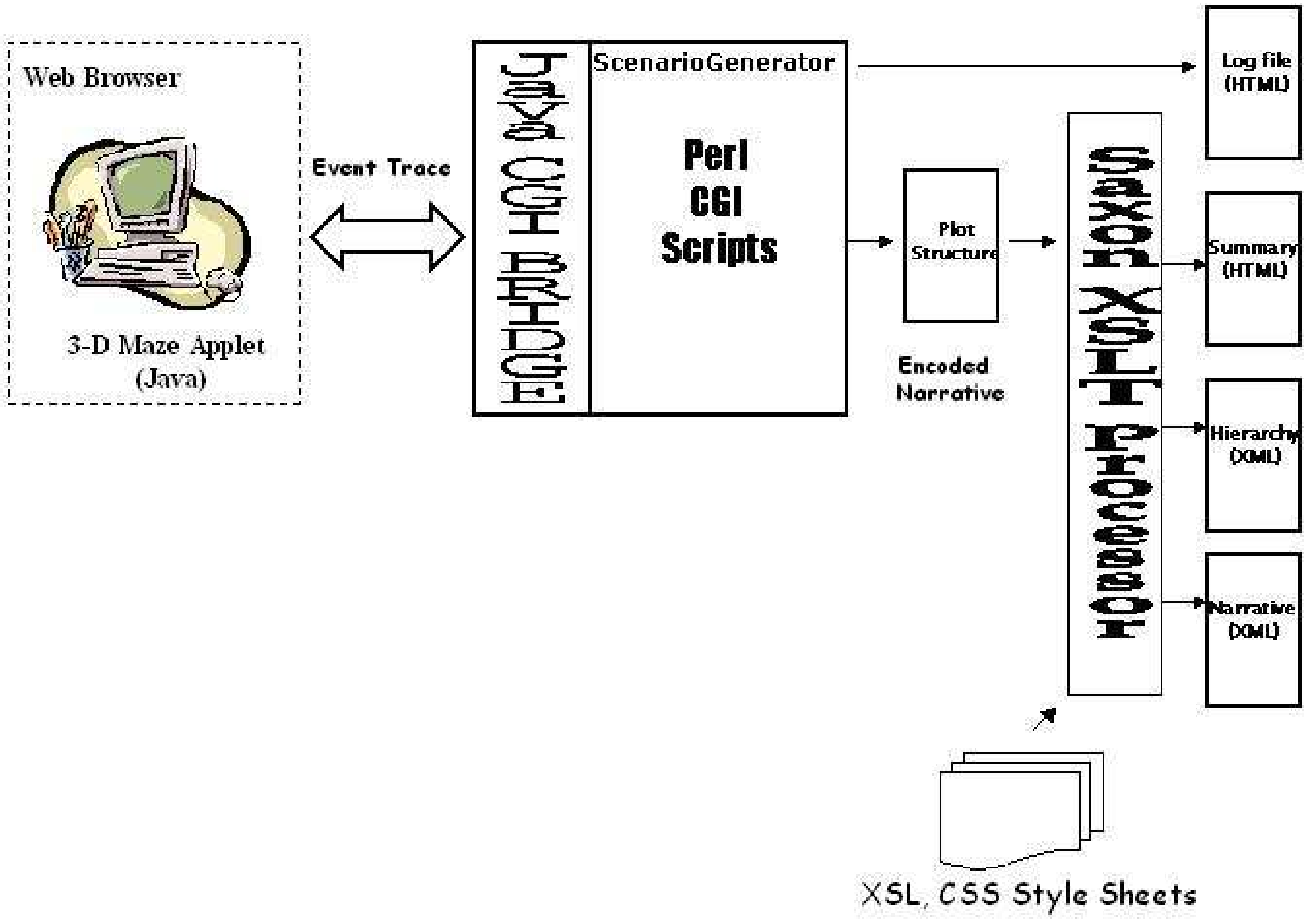


Figure 25: Maze Game Design

The components of Aesop are implementations of the scenario generator architecture as shown in figure 24. Event determination and scenario creation is done with UNIX-style regular expressions for pattern matching. The plot structured, SCML-encoded narrative is output by the CGI scripts. *Saxon* [157], an XSLT processor, interpreted the XSL code and generated the scenario variants.

Aesop displays the maze using texture-mapped images to represent walls and corridors. Figure 26 on page 93 is a screenshot of the Aesop GUI. The main objective of the game is to move through the maze, locating the red, green, and blue square on the floor. To find a square, the player has to be positioned directly over it. The maze game was chosen to reduce the number of potential actions, events, episodes, and scenarios. There are only four actions possible within game: move forward, move back, turn right, and turn left.

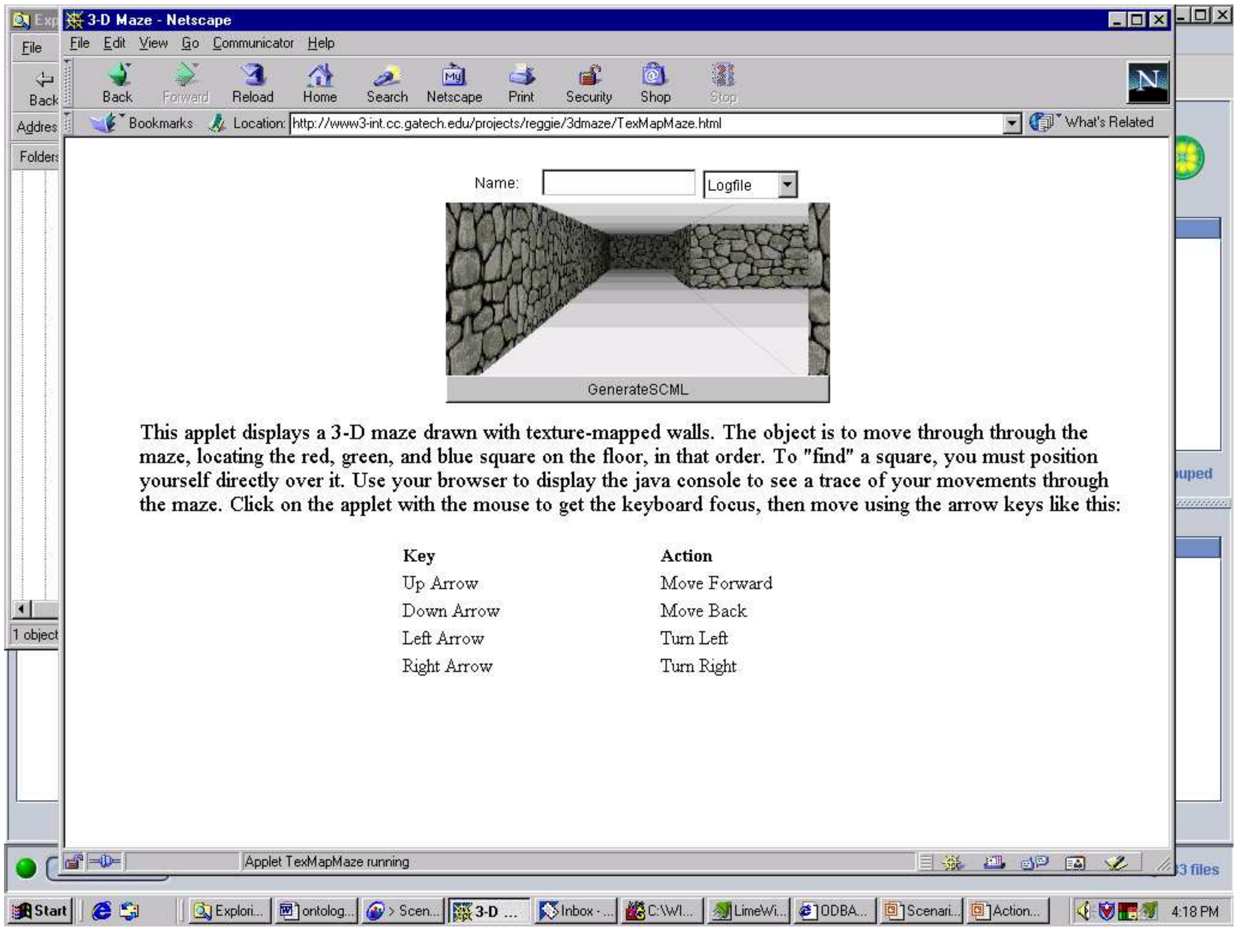
In this simplified example, there are seven possible events as shown in Table 9. The potential episodes are accomplished by achieving the goal of finding a particular square.

Table 9: Events for Scenario Prototype

Entered-The-Maze
Moving-Towards-Wall
Moving-Away-From-Wall
Collides-With-Wall
Found-The-Red-Square
Found-The-Blue-Square
Found-The-Green-Square

The semantic analysis component goes through the log file, collecting actions into events based on these rules for the game:

- Start in Northwest corner of maze, facing due east
- Turning changes direction, not position
- Turning always causes the end of an event
- Move one square at a time
- Four squares or less moving in a direction away from a wall is considered as the moving-away-from-wall event
- If positioned between two walls with less than eight squares between them, use a ratio of half the distance to determine the appropriate event
- Moving more than four squares in one direction is the moving-towards-wall event with respect to the wall the player is facing
- Trying to move forward when in front of a wall is a collision
- A square can be seen from three squares away
- Finding a square occurs when standing on the square

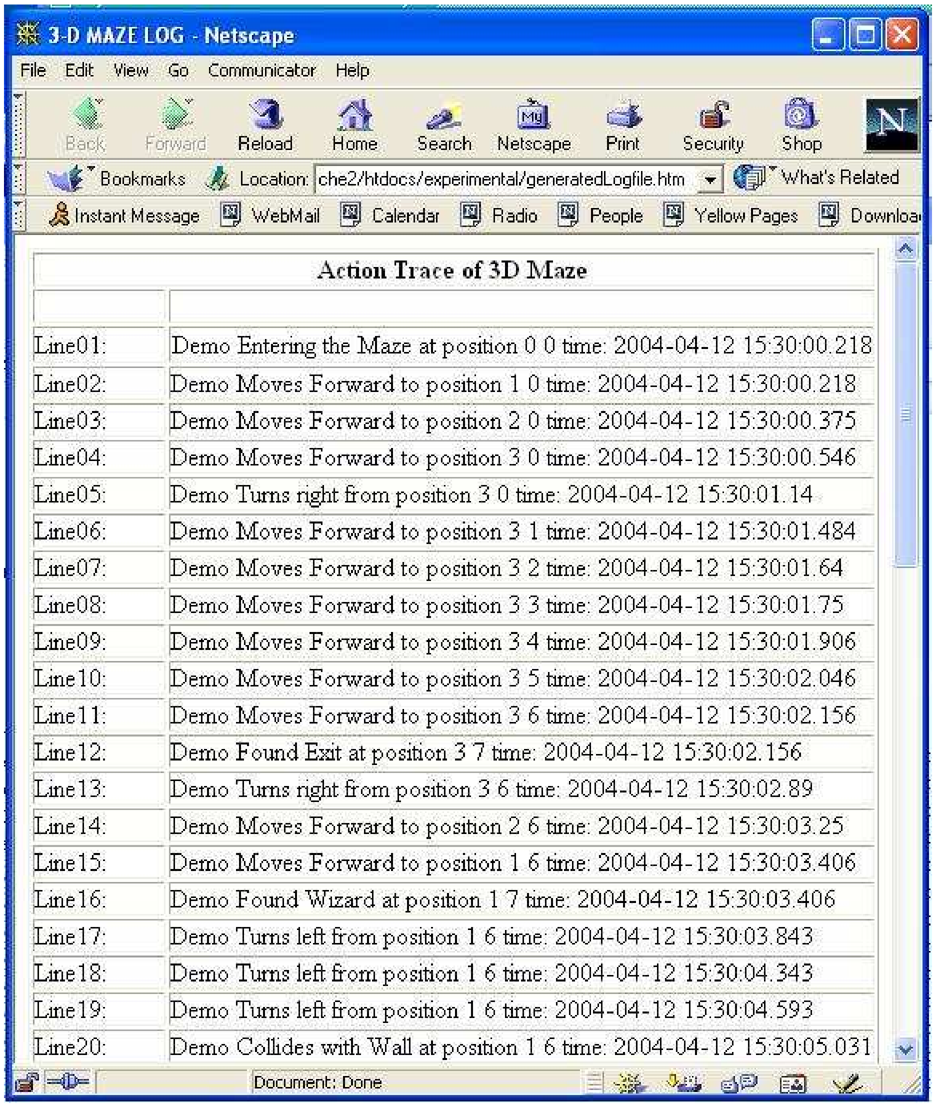

3-D Maze - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks
Location: http://www3-int.cc.gatech.edu/projects/reggie/3dmaze/TexMapMaze.html
What's Related
Name:
Logfile
GenerateSCML
Applet TexMapMaze running
Start
4:18 PM

This applet displays a 3-D maze drawn with texture-mapped walls. The object is to move through through the maze, locating the red, green, and blue square on the floor, in that order. To "find" a square, you must position yourself directly over it. Use your browser to display the java console to see a trace of your movements through the maze. Click on the applet with the mouse to get the keyboard focus, then move using the arrow keys like this:

Key	Action
Up Arrow	Move Forward
Down Arrow	Move Back
Left Arrow	Turn Left
Right Arrow	Turn Right

Figure 26: Aesop GUI screenshot

Using the game rules and performing pattern matching on the events creates the SCML document. The pull-down menu on the maze game browser gives the player the choice of representation of the output. The player enters a name in the text field and selects a format for the output file. The log file choice gives a tabular view of the actions from the maze. The summary view is an HTML file constructed by analyzing the SCML output and transforming it with an XSL style sheet. The hierarchy gives the player a collapsible tree view of the story, shows how actions make up events, how events are separated by scenes, how scenes are contained in episodes, and so on.

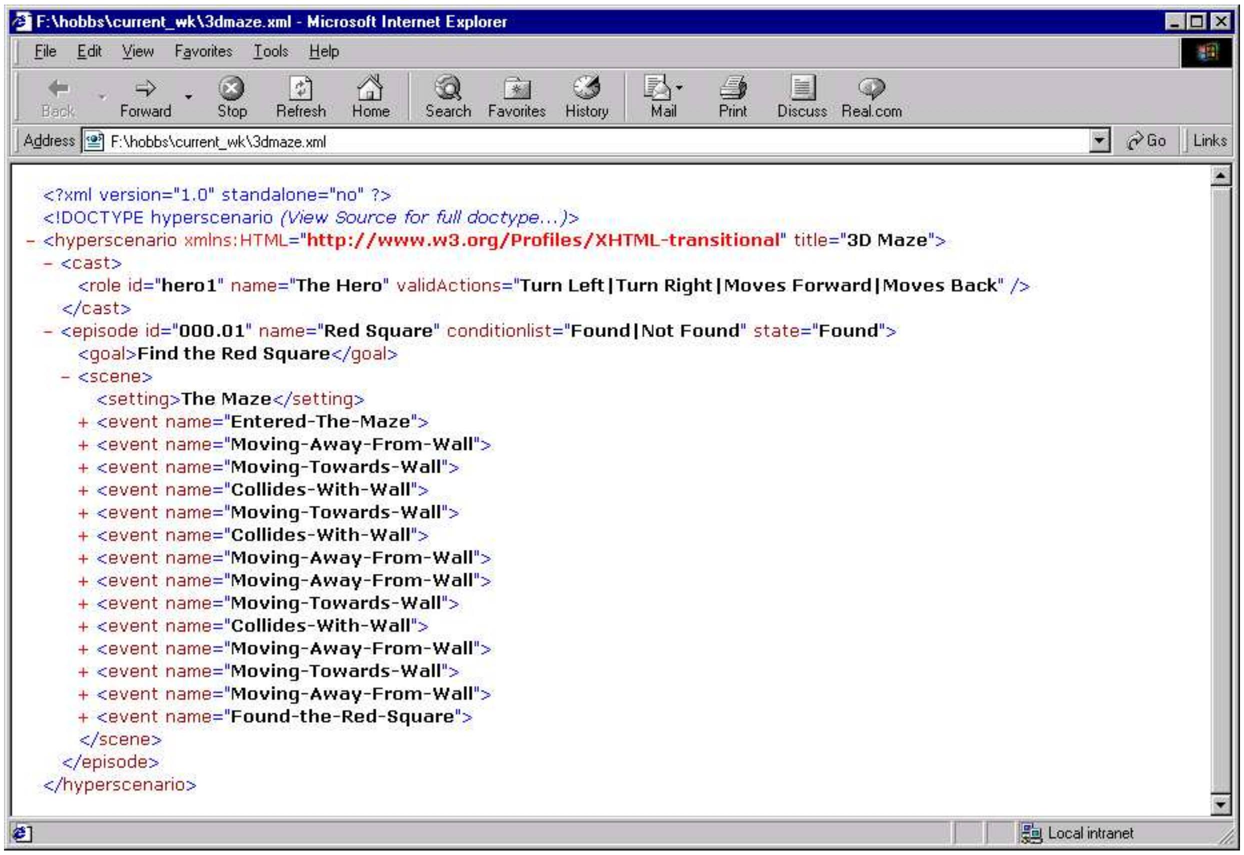

3-D MAZE LOG - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop
Bookmarks Location: che2/htdocs/experimental/generatedLogfile.htm What's Related
Instant Message WebMail Calendar Radio People Yellow Pages
Document: Done

Action Trace of 3D Maze	
Line01:	Demo Entering the Maze at position 0 0 time: 2004-04-12 15:30:00.218
Line02:	Demo Moves Forward to position 1 0 time: 2004-04-12 15:30:00.218
Line03:	Demo Moves Forward to position 2 0 time: 2004-04-12 15:30:00.375
Line04:	Demo Moves Forward to position 3 0 time: 2004-04-12 15:30:00.546
Line05:	Demo Turns right from position 3 0 time: 2004-04-12 15:30:01.14
Line06:	Demo Moves Forward to position 3 1 time: 2004-04-12 15:30:01.484
Line07:	Demo Moves Forward to position 3 2 time: 2004-04-12 15:30:01.64
Line08:	Demo Moves Forward to position 3 3 time: 2004-04-12 15:30:01.75
Line09:	Demo Moves Forward to position 3 4 time: 2004-04-12 15:30:01.906
Line10:	Demo Moves Forward to position 3 5 time: 2004-04-12 15:30:02.046
Line11:	Demo Moves Forward to position 3 6 time: 2004-04-12 15:30:02.156
Line12:	Demo Found Exit at position 3 7 time: 2004-04-12 15:30:02.156
Line13:	Demo Turns right from position 3 6 time: 2004-04-12 15:30:02.89
Line14:	Demo Moves Forward to position 2 6 time: 2004-04-12 15:30:03.25
Line15:	Demo Moves Forward to position 1 6 time: 2004-04-12 15:30:03.406
Line16:	Demo Found Wizard at position 1 7 time: 2004-04-12 15:30:03.406
Line17:	Demo Turns left from position 1 6 time: 2004-04-12 15:30:03.843
Line18:	Demo Turns left from position 1 6 time: 2004-04-12 15:30:04.343
Line19:	Demo Turns left from position 1 6 time: 2004-04-12 15:30:04.593
Line20:	Demo Collides with Wall at position 1 6 time: 2004-04-12 15:30:05.031

Figure 27: Maze Logfile

Figure 27 illustrates the log file created by a game session. This log file is similar to the standard output of a simulation: the position of objects, actions performed, and temporal information. There is no additional context added, just the action trace.

A straightforward transformation of the output is a hierarchical tree structure, as illustrated in Figure 28 on the next page.

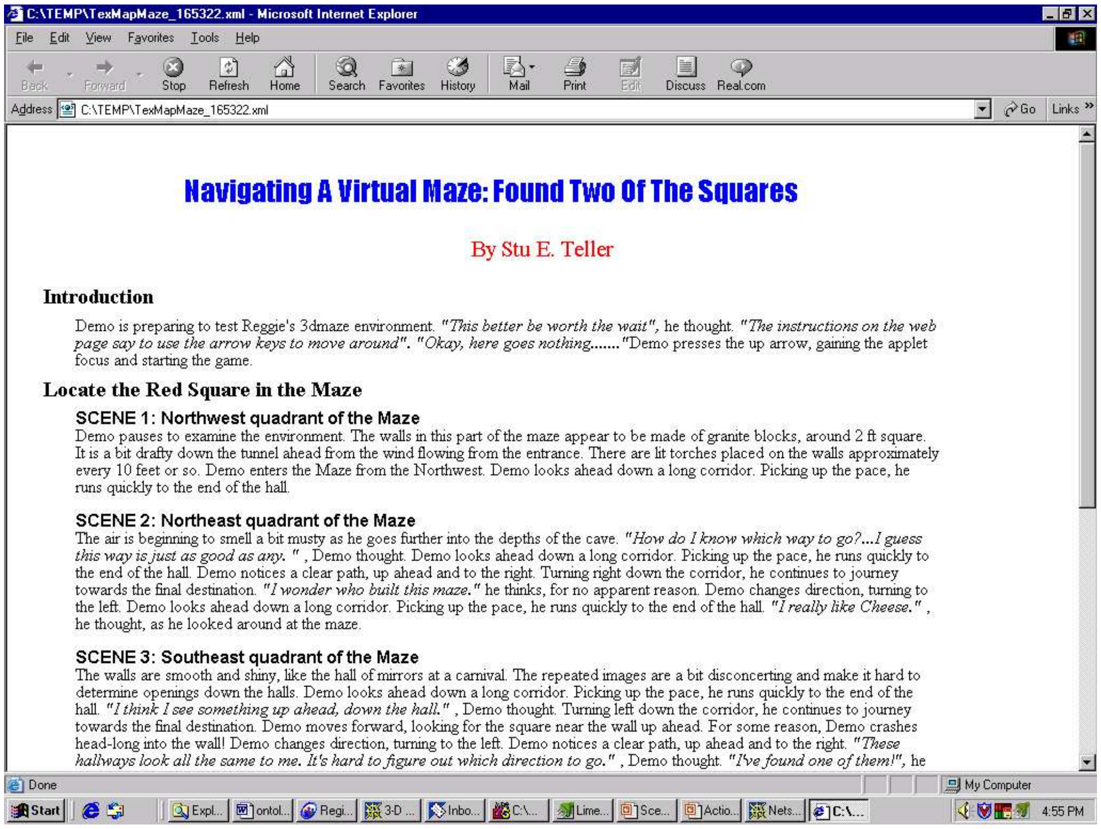


F:\hobbs\current_wk\3dmaze.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Discuss Real.com
Address F:\hobbs\current_wk\3dmaze.xml
Go Links
Local intranet

```
  <?xml version="1.0" standalone="no" ?>
  <!DOCTYPE hyperscenario (View Source for full doctype...)>
- <hyperscenario xmlns:HTML="http://www.w3.org/Profiles/XHTML-transitional" title="3D Maze">
  - <cast>
      <role id="hero1" name="The Hero" validActions="Turn Left|Turn Right|Moves Forward|Moves Back" />
    </cast>
  - <episode id="000.01" name="Red Square" conditionlist="Found|Not Found" state="Found">
      <goal>Find the Red Square</goal>
    - <scene>
        <setting>The Maze</setting>
      + <event name="Entered-The-Maze">
      + <event name="Moving-Away-From-Wall">
      + <event name="Moving-Towards-Wall">
      + <event name="Collides-With-Wall">
      + <event name="Moving-Towards-Wall">
      + <event name="Collides-With-Wall">
      + <event name="Moving-Away-From-Wall">
      + <event name="Moving-Away-From-Wall">
      + <event name="Moving-Towards-Wall">
      + <event name="Collides-With-Wall">
      + <event name="Moving-Away-From-Wall">
      + <event name="Moving-Towards-Wall">
      + <event name="Moving-Away-From-Wall">
      + <event name="Found-the-Red-Square">
      </scene>
    </episode>
  </hyperscenario>
```

Figure 28: SCML Scenario Hierarchy

One of the most interesting SCML document transformations is that of a literary narrative format. The narrative reflects the actions that were done in the game simulation, with extra text around it to make up a story. Each time the game is played, depending upon the scenario variant and the actions, a different story is generated. The text of the narrative is formatted using XSL and *Cascading Style Sheets (CSS)* [162] to render a web page, as seen in Figure 29. Aesop can detect particular sequences of events, like running into the walls repeatedly, and generate appropriate text to describe what happened. The counterpart of this technique in the modeling and simulation community is the addition of army doctrinal information or *Standard Operating Procedures (SOPs)* around known tasks.


C:\TEMP\TexMapMaze_165322.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Real.com
Address C:\TEMP\TexMapMaze_165322.xml
Go Links
Done
My Computer
Start
4:55 PM

Navigating A Virtual Maze: Found Two Of The Squares

By Stu E. Teller

Introduction

Demo is preparing to test Reggie's 3dmaze environment. *"This better be worth the wait"*, he thought. *"The instructions on the web page say to use the arrow keys to move around"*. *"Okay, here goes nothing.......*"Demo presses the up arrow, gaining the applet focus and starting the game.

Locate the Red Square in the Maze

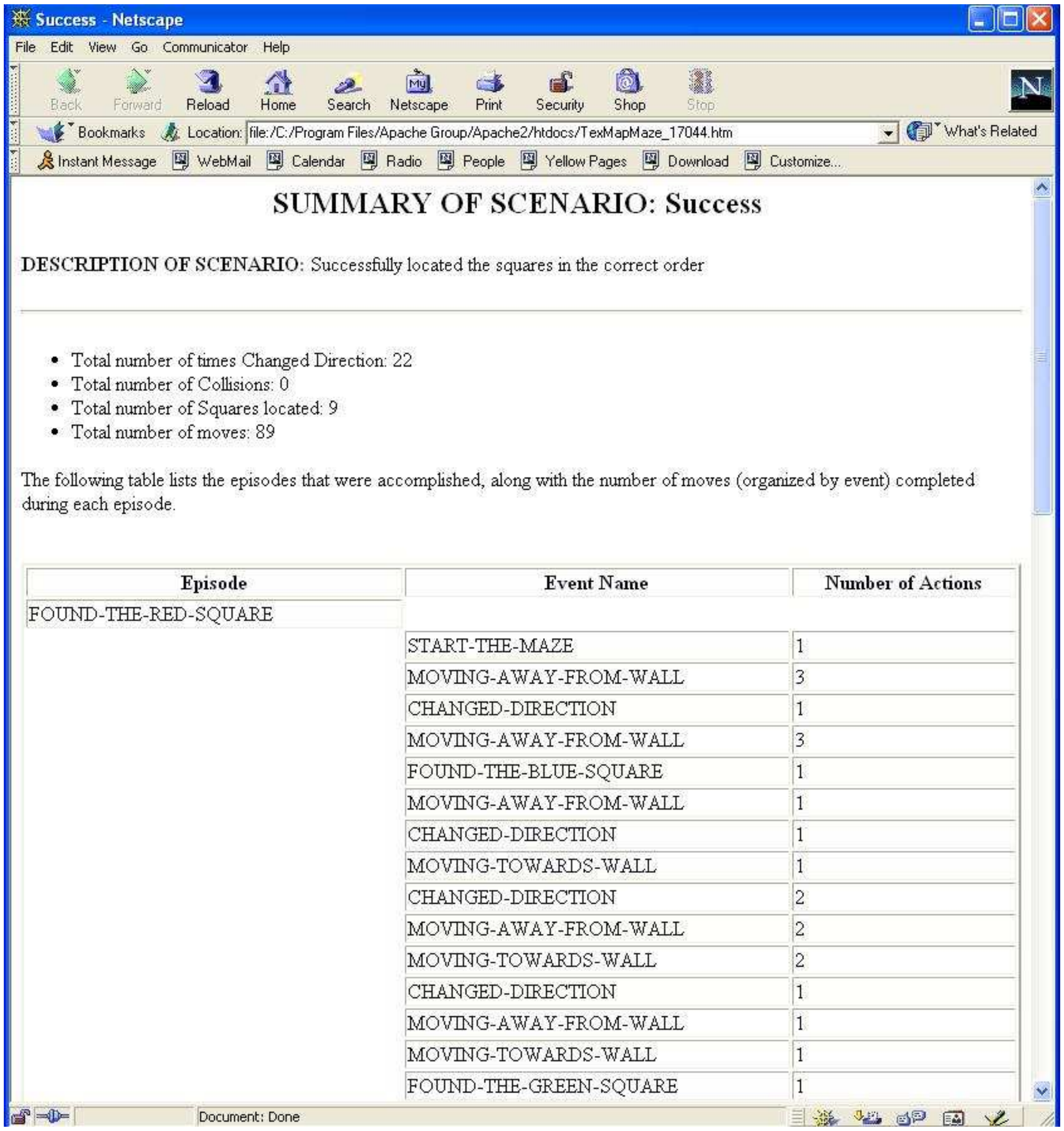
SCENE 1: Northwest quadrant of the Maze
Demo pauses to examine the environment. The walls in this part of the maze appear to be made of granite blocks, around 2 ft square. It is a bit drafty down the tunnel ahead from the wind flowing from the entrance. There are lit torches placed on the walls approximately every 10 feet or so. Demo enters the Maze from the Northwest. Demo looks ahead down a long corridor. Picking up the pace, he runs quickly to the end of the hall.

SCENE 2: Northeast quadrant of the Maze
The air is beginning to smell a bit musty as he goes further into the depths of the cave. *"How do I know which way to go?...I guess this way is just as good as any. "* , Demo thought. Demo looks ahead down a long corridor. Picking up the pace, he runs quickly to the end of the hall. Demo notices a clear path, up ahead and to the right. Turning right down the corridor, he continues to journey towards the final destination. *"I wonder who built this maze."* he thinks, for no apparent reason. Demo changes direction, turning to the left. Demo looks ahead down a long corridor. Picking up the pace, he runs quickly to the end of the hall. *"I really like Cheese."* , he thought, as he looked around at the maze.

SCENE 3: Southeast quadrant of the Maze
The walls are smooth and shiny, like the hall of mirrors at a carnival. The repeated images are a bit disconcerting and make it hard to determine openings down the halls. Demo looks ahead down a long corridor. Picking up the pace, he runs quickly to the end of the hall. *"I think I see something up ahead, down the hall."* , Demo thought. Turning left down the corridor, he continues to journey towards the final destination. Demo moves forward, looking for the square near the wall up ahead. For some reason, Demo crashes head-long into the wall! Demo changes direction, turning to the left. Demo notices a clear path, up ahead and to the right. *"These hallways look all the same to me. It's hard to figure out which direction to go."* , Demo thought. *"I've found one of them!"*, he

Figure 29: Narrative generated from simulation trace

Other transformations have been coded for the simulation. A tabular summary file in HTML format can be created to eliminate the details for a high-level view of the story (Figure 30). The text from the literary narrative can be further transformed by re-direction through a translation tool such as babelfish [109] to represent the story in other languages. Existing XML technologies, such as VoiceXML [167], can be used to generate an audio version of the story with the help of an appropriate text-to-speech engine.


Success - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: file:/C:/Program Files/Apache Group/Apache2/htdocs/TexMapMaze_17044.htm
What's Related
Instant Message WebMail Calendar Radio People Yellow Pages Download Customize...
Document: Done

SUMMARY OF SCENARIO: Success

DESCRIPTION OF SCENARIO: Successfully located the squares in the correct order

- Total number of times Changed Direction: 22
- Total number of Collisions: 0
- Total number of Squares located: 9
- Total number of moves: 89

The following table lists the episodes that were accomplished, along with the number of moves (organized by event) completed during each episode.

Episode	Event Name	Number of Actions
FOUND-THE-RED-SQUARE		
	START-THE-MAZE	1
	MOVING-AWAY-FROM-WALL	3
	CHANGED-DIRECTION	1
	MOVING-AWAY-FROM-WALL	3
	FOUND-THE-BLUE-SQUARE	1
	MOVING-AWAY-FROM-WALL	1
	CHANGED-DIRECTION	1
	MOVING-TOWARDS-WALL	1
	CHANGED-DIRECTION	2
	MOVING-AWAY-FROM-WALL	2
	MOVING-TOWARDS-WALL	2
	CHANGED-DIRECTION	1
	MOVING-AWAY-FROM-WALL	1
	MOVING-TOWARDS-WALL	1
	FOUND-THE-GREEN-SQUARE	1

Figure 30: Maze Summary Web Page

4.2 SCML-based Decision Support for Military Planning

Because of its utility for decision-making in general, a scenario-centered approach can be applied to other problem solving domains, such as military planning using modeling and simulation. The difficulty in developing a meta-model for simulations is due to the variations in styles, categories, and implementations of simulations. All simulations are essentially stories. They are stories about what was done, what is being done, or what can be done. These different forms of narrative can be used for problem solving, training, entertainment, and other decision-oriented activities.

Constructive simulations, the man-in-the-loop style of simulation, are being used by the Army to determine the affect of new technologies on *Command, Control, Communications, Computers, and Intelligence (C4I)* [175]. These simulations are generally done as war games in which participants are given assigned roles. The *Battle Management Language (BML)* effort being organized by the *Simulations-to-C4I Overarching Integrated Product Team (SIMCI OIPT)* is an attempt to solve simulation interoperability problems [21]. BML's approach is to standardize the terminology and symbology based on doctrinal information: to allow simulations a common way of interacting. This effort is primarily in the exploration stage, with no specification or implementation determined. There has been discussion in the C4I community on implementing BML in one of the Army's tactical databases to make it available to different representations, particularly XML.

The largest such effort for interoperability revolves around the *High-Level Architecture (HLA)* [38] standards development. The HLA was developed to create a common architecture applicable across different types of simulation environments. The HLA Working Group, governed by both the IEEE Computer Society and the *Simulations Interoperability Standards Committee (SISC)*, has developed three draft standards:

1. Framework and Rules (IEEE P1516) – which describes the responsibilities for simulations that adhere to the standard and the format for a *Simulation Object Model (SOM)* [181].

2. Federate Interface Specification (IEEE P1516.1) – describes the interactions between

simulations and a Run-time Infrastructure [94].

3. *Object Model Template (OMT)* Specification (IEEE P1516.2) – which prescribes the objects, attributes, interactions, and parameters that are required for a SOM [180].

It is the OMT that is critical for describing the format of data exchanged between simulations. An HLA SOM is a specification of the intrinsic capabilities that an individual simulation could provide to HLA-compliant federations [180]. The OMT *Data Interchange Format (DIF)* supplies a structure that could be used by automated tools to convert from one representation of an OMT to another. Initially, the OMT DIF was represented primarily in BNF notation. There is now available an XML version of the OMT that will leverage many existing tools and techniques for interchange.

The OMT is useful in describing the low-level attributes of data objects that could be shared by simulations, but does not address how to map these objects into varying environments. In order to share higher-level information between simulations, it is necessary to describe the context under which SOMs are manipulated within each simulation. Interoperability at this level requires meta-models, adaptive models, and common repositories [174]. The *Model-Driven Architecture (MDA)* [125] initiative has been introduce to create specifications based on formal models. Under the MDA, these models could take advantage of technology such as the XML Metadata Interchange (XMI) specification [71], *Simple Object Access Protocol (SOAP)* [15], and other standards.

4.2.1 Simulation-to-C4I Interoperability

There has been much discussion on the creation of a common framework for integrating information from *Modeling and Simulations (M&S)* applications with data generated by existing C4I Systems [175]. Both domains are shifting towards distributed, network-centric approaches for designing, implementing, and manipulating scenarios for the war fighter. The *Joint Technical Architecture (JTA)* [104] proposes standards for C4I interoperability while the *Extensible Modeling and Simulation Framework (XMSF)* [18] promotes the use of web services, such as XML and SOAP, as M&S standards. In order to share semantically useful data from such a diverse range of systems, it is necessary to describe a conceptual model that can map the rules and behaviors in an abstract way. The use of narrative, in the form of scenarios, makes this representation possible. This narrative structure can be derived from simulation data to generate canonical stories from domain information.

The Hyperscenario Framework was first presented to the *Simulations Interoperability Standards Organization (SISO)* community during the Spring 2003 SIW conference [86]. There were two factors that made the work particularly unique: 1) the ideas and concepts originated from the software engineering community and 2) its focus on using a narrative meta-model not associated with any operational military environment. However, it was deemed applicable by the modeling and simulation (M & S) community, as it presented an alternative approach for simulations interoperability. There are a myriad of standards being developed that address simulation interoperability at the object level. To support an exchange of semantics from a simulations tool to decision-oriented C4I systems, it is the context that must be shared. What must be transferred from simulation to C4I is not where objects are in the virtual battlefield, but how and under what circumstances did they get there. Conference participants expressed a high level of interest in issues concerning M&S and C4I interoperability during the previous SIW conference. Dr. Andreas Tolk, a leading M&S researcher, gave a widely attended presentation that focused on the how and the why of creating a framework for the interchange of information between the two domains [175]. Tolk discussed the necessity for such a framework being driven by the increasing scope of applications. Many of the systems are vertically integrating with higher level systems, with little support for cross-pollination between environments. There are also emerging requirements for embedded training simulations and the ability to immediately use simulation data under operational conditions. A common framework requires the use of web services to support network-centric applications, the establishment of direct working relationships with the C4I community, and the development of components to support the interaction.

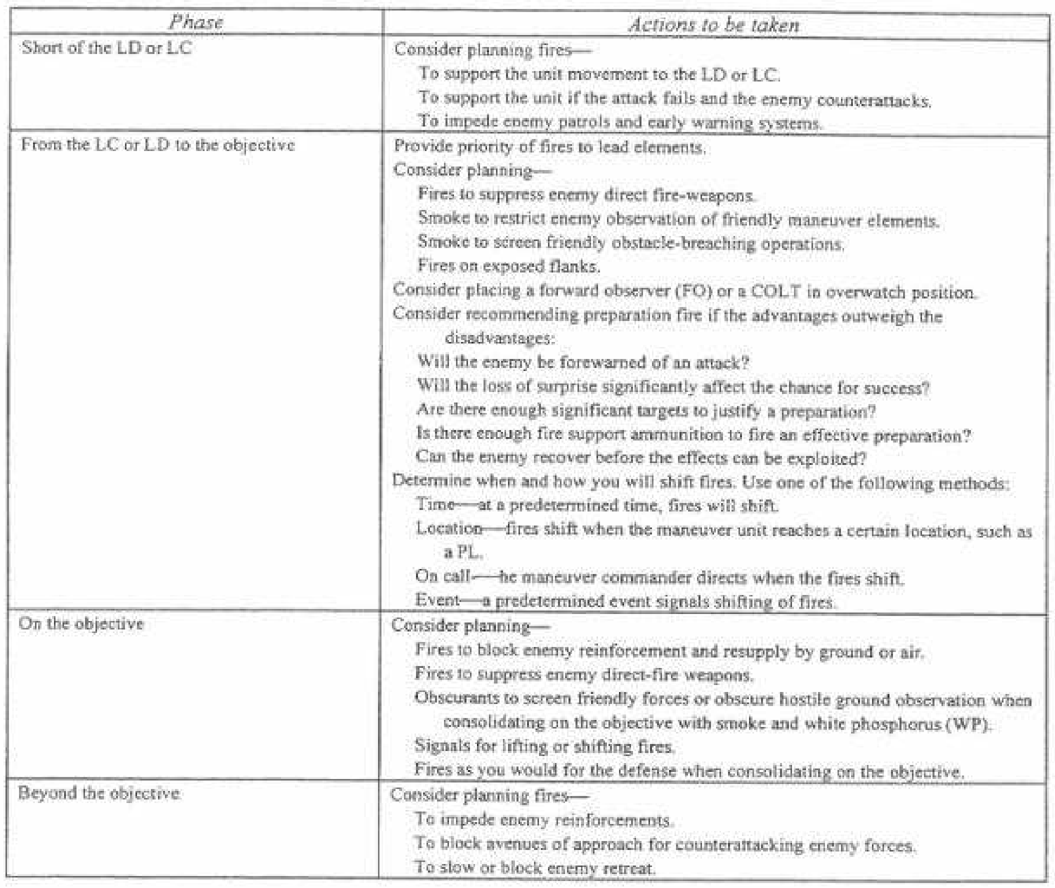
Another approach was the use of the *Common Operating Environment (COE)*, developed by the *Defense Information Systems Agency (DISA)* as a model for interoperability [174]. The COE is an architectural approach towards software development that focuses on reusable software and the establishment of guidelines. There are over one hundred systems across the DoD that adhere to the guidelines and policies of COE. But the M&S community lacks a central governing institution on the level of DISA to administer and mandate an infrastructure like COE. However, there are a large amount of lessons learned on interoperability issues that could be gleaned from examining the COE process.

The Hyperscenario Framework and SCML can be used to communicate information from a simulation environment to a C4I system. The challenge is in determining a mapping of story elements from one domain to the other. A characteristic that represents a single action in a simulation may be operationalized as several events in a decision support system. This is where the ability to add annotations, external documentation, and other forms of narrative become of vital importance. The simulation environment contains objects with attributes and behaviors. There may be some low-level context available within the system, but only what could be inferred based on the constraints of the software. For example, if the simulation contained tanks or armored personnel carriers, it would be fairly straightforward to determine their location in the battlefield and their current mission status. But why they are positioned the way they are and under what circumstances would not be apparent. The next section contains a brief example of how the information could be encoded from a simulation in order to be useful in a command and control situation. The information in the example was not actually used with either system, but is presented here for the purpose of illustration.

The *One Semi-Autonomous Forces (OneSAF)* Objective System is a simulations environment being designed to support a wide-range of military training and planning operations [79]. The fundamental components are behaviors, physical models, and behavior agents. The behaviors contained in OneSAF are low-level functions based on doctrine. These behaviors control the use of the physical models in the system, such as weapons, sensors, and communications. The behavior agents are used for planning and execution; these are the command and control agents that manipulate the behaviors. These building block components are used to create composite behaviors that are more complex, entities that represent equipment and personnel, and units that represent organizational structures, such as platoons and battalions.

An example composite behavior would be *MoveAndShoot*. The definition for this behavior includes a tabular list of variables, an execution timeline, and the primitive behaviors involved. Within *MoveAndShoot*, the primitive behaviors are: moving around, sensing the environment, shooting the weapon system, and reporting the status. Composite behaviors such as this one could be associated with the episodic elements in the scenario ontology, because they are goal-based. Each of the primitive behaviors that are contained within a composite are SCML events; series of actions that may or may not support a subgoal but are vital to completing an episode.

The Hyperscenario Framework is extremely well-suited for the battle planning domain, because the Army has policies and procedures for every task and activity necessary for its operations. There is even a procedure for the decision-making process, called the *Military Decision to Making Process (MDMP)* [178]. This process outlines seven separate steps to be performed for battle planning: 1) receipt of mission, 2) mission analysis, 3) *COA (Course of Action)* development, 4) COA analysis, 5) COA comparison, 6) COA approval, and 7) orders production. The seven steps lend themselves to representation as episodes within a scenario. Within each step, there are set actions to be performed with specific responsibilities for particular participants. These actions and responsibilities map well to the concept of actors, stakeholders, actions, and events. There are *Tactics, Techniques, and Procedures (TTP)* that govern how the battle should unfold. The episodic arrangements of these phases make it easier to develop battlefield templates to represent how the Army fights.



Phase	*Actions to be taken*
Short of the LD or LC	Consider planning fires— To support the unit movement to the LD or LC. To support the unit if the attack fails and the enemy counterattacks. To impede enemy patrols and early warning systems.
From the LC or LD to the objective	Provide priority of fires to lead elements. Consider planning— Fires to suppress enemy direct fire-weapons. Smoke to restrict enemy observation of friendly maneuver elements. Smoke to screen friendly obstacle-breaching operations. Fires on exposed flanks. Consider placing a forward observer (FO) or a COLT in overwatch position. Consider recommending preparation fire if the advantages outweigh the disadvantages: Will the enemy be forewarned of an attack? Will the loss of surprise significantly affect the chance for success? Are there enough significant targets to justify a preparation? Is there enough fire support ammunition to fire an effective preparation? Can the enemy recover before the effects can be exploited? Determine when and how you will shift fires. Use one of the following methods: Time—at a predetermined time, fires will shift. Location—fires shift when the maneuver unit reaches a certain location, such as a PL. On call—he maneuver commander directs when the fires shift. Event—a predetermined event signals shifting of fires.
On the objective	Consider planning— Fires to block enemy reinforcement and resupply by ground or air. Fires to suppress enemy direct-fire weapons. Obscurants to screen friendly forces or obscure hostile ground observation when consolidating on the objective with smoke and white phosphorus (WP). Signals for lifting or shifting fires. Fires as you would for the defense when consolidating on the objective.
Beyond the objective	Consider planning fires— To impede enemy reinforcements. To block avenues of approach for counterattacking enemy forces. To slow or block enemy retreat.

Figure 31: Battle Planning Phases

Figure 31 is a chart that represents a series of phases of the battle (based on location on the battlefield) and appropriate action to be taking based on standard Army doctrine. In the diagram, *LD* means *line of departure* and *LC* means *line of contact*. The table describes how to play an offensive strategy based on a switch of scenes, from before encountering the enemy to the initial contact and afterwards. As if this field manual and others like it are not detailed enough, the military has developed what it refers to as the *Universal Task List (UTL)* [179] to consolidate all the actions and responsibilities for every possible operation.

The C4I environment for this example is the *Situational Awareness (SA)* requirements analysis effort for *Military Operations on Urbanized Terrain (MOUT)* [50]. The purpose of this SA research was to use a goal-directed task analysis and subject matter experts to determine SA requirements during MOUT missions. Military officers participated in a series of simulated missions and discussed the SA requirements during halts in the mission. By analyzing the responses, the researchers developed set of measurement tools to assess situational awareness. A useful output of the research was the development of a detailed goal-hierarchy that represented the MDMP [178]. This hierarchy (Figure 32) was developed using doctrinal information, army regulations, input from the subject matter experts and goal decomposition techniques. Because episodes in the scenario conceptual model are goal-based, this becomes the bridge to establish an automated connection between OneSAF and the SA task.

The goal-hierarchy development by the SA environment can be used to place additional data together with the behavioral data generated by OneSAF. This additional information can be hyper-linked into the documents and includes descriptions of the force structure, overlays from the battlefield, and images for battle damage assessment.

Figure 33 is an excerpt from a *concept of operation* document[1] . This scenario describes a battle from the perspective of an army tank platoon. In this scenario, each of two platoons has four M1 tanks. The enemy also has two platoons, each of those contain three T72 Russian tanks. This scenario information can be used to generate orders and describe mission objectives for OneSAF. The participants are expected to know the particulars of how the objectives are to be met.

[1]This is from a mission scenario developed at West Point, originally used for ModSAF.

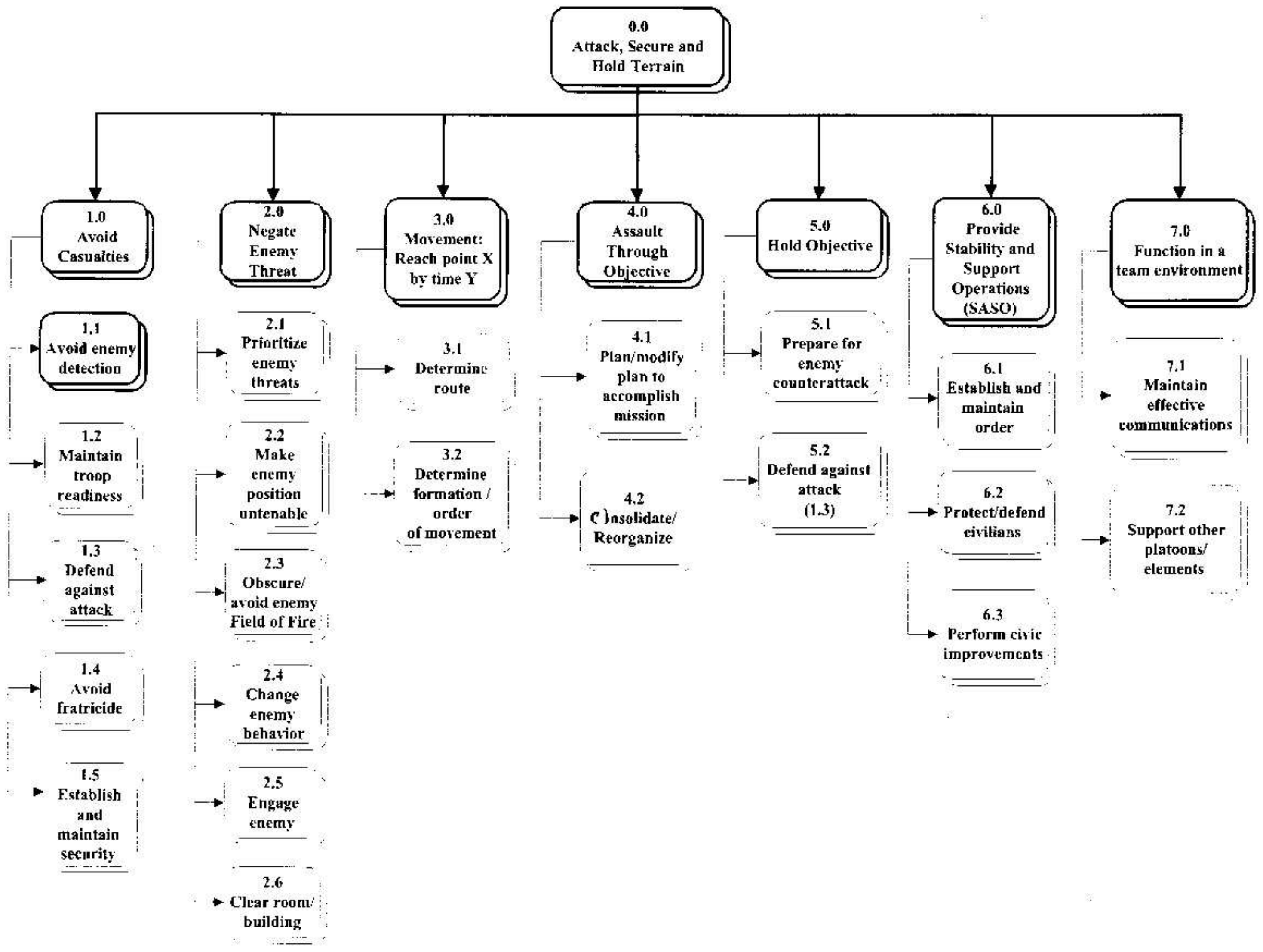

0.0 Attack, Secure and Hold Terrain
1.0 Avoid Casualties
2.0 Negate Enemy Threat
3.0 Movement: Reach point X by time Y
4.0 Assault Through Objective
5.0 Hold Objective
6.0 Provide Stability and Support Operations (SASO)
7.0 Function in a team environment
1.1 Avoid enemy detection
1.2 Maintain troop readiness
1.3 Defend against attack
1.4 Avoid fratricide
1.5 Establish and maintain security
2.1 Prioritize enemy threats
2.2 Make enemy position untenable
2.3 Obscure/ avoid enemy Field of Fire
2.4 Change enemy behavior
2.5 Engage enemy
2.6 Clear room/ building
3.1 Determine route
3.2 Determine formation / order of movement
4.1 Plan/modify plan to accomplish mission
4.2 Consolidate/ Reorganize
5.1 Prepare for enemy counterattack
5.2 Defend against attack (1.3)
6.1 Establish and maintain order
6.2 Protect/defend civilians
6.3 Perform civic improvements
7.1 Maintain effective communications
7.2 Support other platoons/ elements

Figure 32: Situational Awareness Goal Hierarchy

```
* First Platoon (4 M1s):
1st PLT will occupy attack position vic. ntc235890 and orient
NNE in order to prevent enemy movement west along roadway vic.
ntc240905. If contact is made, 1st PLT will conduct an ambush
with 2nd PLT in support to the west at ntc221901. 1st PLT has
priority of fire with 155 howitzer unit in support of
operation. An Artillery-laid minefield is in place on roadway
at ntc238905 to assist in fixing the enemy. Upon neutralizing
the enemy, 1st PLT will assault to OBJ Red and set up a hasty
perimeter.
* Second Platoon (4 M1s):
2nd PLT will occupy attack position vic. ntc221901 and orient
NE in order to prevent enemy movement west along roadway vic.
ntc240905.
* Enemy (2 platoons; 3 T72Ms each):
Move along Rt. T72M. Upon detecting enemy PLTs, enemy platoons
will conduct contact drills and return fire.
```

Figure 33: Concept Of Operations

Figure 34 is an example of how the SCML formatted battle scenario looks. Once the narrative of the battle is in this format it can be fed into other scenario applications to improve situational awareness, generate after-action reports, perform COA analysis, generate operational orders, or be inserted into a lessons-learned database to support focused searches.

```
<?xml version="1.0"?>
<!-- Revision: 1.0 /usr/stb/hobbs/scml/ntcplatoon.scml -->
<!DOCTYPE hyperscenario SYSTEM "c:/current_wk/scml.dtd">
<hyperscenario title="NTC Platoon Battle">
<purpose>Maneuver Battle exercise for NTC tank platoon</purpose>
<role name="2nd Platoon" id="2 PLT Charlie Company">
<cast>
<actor>M1 Tank Squad #1</actor>
<actor>M1 Tank Squad #2</actor>
<actor>M1 Tank Squad #3</actor>
<actor>M1 Tank Squad #4</actor></role>
</cast>
<title>NTC Platoon Operations</title>
<setting>ModSAF Fort Irwin Desert Terrain</setting>
<script>
<episode name="Movement to Contact">
<description>Maneuver conducted to develop the situation and
to establish or regain contact</description>
<goal>Establish contact with OPFOR</goal>
<action></action>
</episode>
<episode name="Countermobility">
<description>The construction of obstacles and emplacement of
mine fields to delay, disrupt, and destroy the enemy by
reinforcement of the terrain. The primary purpose of countermobility
operations is to slow or divert the enemy,to increase time for target
acquisition, and to increase weapon effectiveness. See FMs 3-50,3-100,
5-102, and 5-250</description><goal>Slow or Divert the OPFOR</goal>
</episode>
<episode name="Defend in Sector">
<description>A technique that requires a defending unit to prevent
enemy forces from passing beyond the rear boundary of the sector while
retaining flank security and ensuring integrity of effort within the
parent unit's scheme of maneuver. Initial positions generally are
established as far forward as possible, but a commander may use any
technique to accomplish the mission. The higher commander will normally
assign no-penetration criteria. See FMs 7-30,71-100,71-123,100-5,and
100-15.</description>
</episode>
```

Figure 34: SCML-Encoded Battle Scenario

4.2.2 Semantic Interoperability using DAML+OIL

In [52], Horrocks et. al. describe ontologies as shared and common understandings of a domain communicated between people and heterogeneous application systems. In philosophy, an ontology is defined as a theory about the nature of existence. An ontology can be considered as a set of concepts, classes, properties, and behaviors from a domain. An interesting analogy proposed by Horrocks is that ontologies are to schemas what *Entity Relationship Diagrams (ERDs)* are to relational databases. Ontologies and ERDs are used to model theory, in this case, domain theory and database models, respectively. *XML-Schemas (XMLS)* [189] and relational databases are particular implementations of the theories. The analogy breaks down in that ontologies are much more expressive the ERD, allowing for axiomatic and formal definitions of concepts/classes. The OIL language was an effort by European researchers to create a computational framework for ontologies that: 1) provides modeling primitives from frame based and *Descriptive Logic (DL)*-based ontologies [9], 2) had consistent semantics, and 3) allowed for automated reasoning support.[52] An ontology in OIL consists of a container and definitions. OIL definitions include imports (which support the inclusion of external OIL modules), a rule-base of axioms and global constraints, and class definitions [136].

DAML, DAML-ONT, and *Resource Description Framework (RDF) [171]* were efforts by US researchers and the government to support the creation of the Semantic Web [91]. Tim Berners-Lee, the inventor of the Worldwide Web, envisions the Semantic Web as the availability of resources to automate processes accomplished through semantic markup and metadata annotations describing content and functionality. These automated processes, called *software agents*[2] , are programs that can form connections and communicate with other programs. In [88], Holmes and Kogut describe agents as the next generation user interface consisting of abstract components that have goals, can interact with humans through speech acts, and share a context for efficient communication. The purpose of DAML and

[2]sometime referred to as intelligent agents

DAML-ONT was to support software agents by extending XML and RDF. RDF is a standard to represent metadata for web resources using an XML syntax. The *RDF Schema language (RDFS)* incorporates object-oriented concepts such as classes and attributes. Using RDFS syntax, DAML can represent ontologies and markup web resources with links between ontologies.

The problem for the semantic web is that DAML, with its RDFS syntax, cannot describe services with the necessary level of detail. DAML is excellent for content description and RDFS does qualify as an ontology language. However, the requirements for a web ontology language are: 1) compatibility with existing standards (XML and RDF), 2) ease of use with standard *Knowledge Representation (KR)* idioms, 3) expressive formal specification, and 4) support for automated reasoning [90]. The merging of the two research efforts into *DARPA Agent Markup Language+Ontology Inference Layer (DAML+OIL)*[3] was intended to support these requirements.

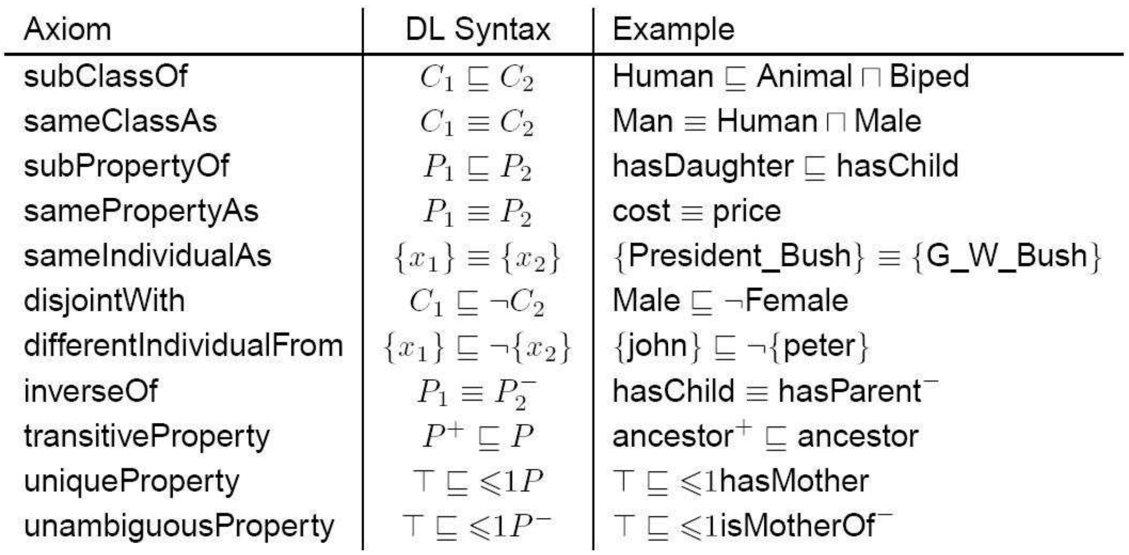
Figure 35 is an example of DAML+OIL syntax. The RDFS primitives, such as classes and unions, are still present in DAML+OIL. Class membership and property definitions can be shown in the RDF write up. This example describes restrictions on the class *Person* on the property *hasChild* with respect to the class *Doctor.*

[3]The DAML+OIL framework is being renamed OWL-S. (Ontology Web Language)

```
<daml:Class>
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Person"/>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasChild"/>
      <daml:toClass>
        <daml:unionOf rdf:parseType="daml:collection">
          <daml:Class rdf:about="#Doctor"/>
          <daml:Restriction>
            <daml:onProperty rdf:resource="#hasChild"/>
            <daml:hasClass rdf:resource="#Doctor"/>
          </daml:Restriction>
        </daml:unionOf>
      </daml:toClass>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>
```

Figure 35: DAML+OIL Syntax[4]

The inclusion of OIL axioms adds a richer class of ontological primitives to DAML, such as transitivity, disjointedness, etc. Figure 36 is a list of some of the axioms available in DAML+OIL. Column one is the name of the axiom. The middle column in the table is the DL syntax for describing the rule. Column three is a specific instance of the axiom, for purposes of illustration. In the example syntax of figure 35 the *toClass*, *unionOf*, and *hasClass* axioms are visible[90].



Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$isMotherOf$^-$

Figure 36: DAML+OIL Axioms

The modeling and simulation community has focused on increasing the level of interoperability between systems. Simulations need to not only support the vertical sharing of data, but horizontal interchange between alternate application domains. The HLA establishes guidelines for constructing shareable simulations objects. To share higher-level information between simulations, it is necessary to describe the context under which they are manipulated in a predictable format. Interoperability at this level requires meta-models, adaptive models, and common repositories [174]. For computer-generated forces to be examined at both the object level and the abstract level, an intermediate form to encode behavior is needed. The state of the simulation objects should be associated with the context under which it came about. This context can be introduced through the use of narrative. The narrative structure can be derived from simulation data to generate canonical stories from domain information. By using military doctrine to describe the scenarios that occur within a simulation, the lessons learned can be applied more directly to C4I systems. The hyperscenario framework can be used to capture semantics within computer-generated forces. Scenarios from a simulation platform, such as OneSAF, could then be directly mapped to DAML+OIL. These scenarios are the basis for decision-making; strategic planning, training, lessons-learned repositories, and systems development. (Figure 37)

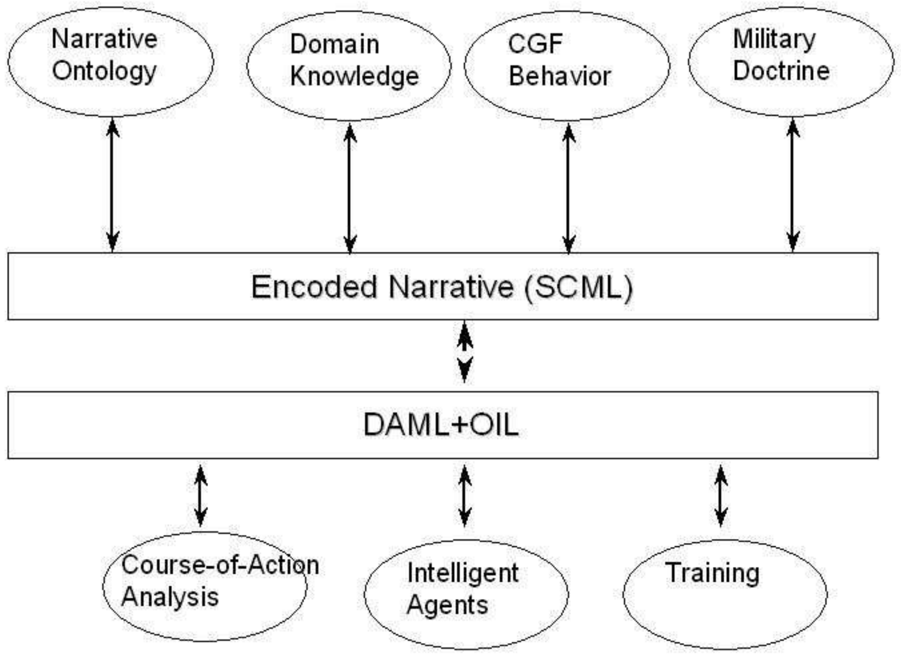


Figure 37: From Narrative to Decision Support

4.3 Mapping SCML to software behavioral design patterns

Software designers have used design patterns and frameworks to describe reusable architectures for development. The approach is to capture the best practices in software engineering to create a catalog of solutions for the practitioner. Incorporating additional semantics within the problem/solution space can further enhance these pattern descriptions. This context can be inserted using narrative. Behavioral design patterns have aspects that can be related to narrative. SCML can be merged with representations of design patterns, which describe the static, dynamic, and collaborative attributes of software solutions. This enables pattern descriptions to be narrative exemplars that could be used directly in design tasks.

A *design pattern* is a description of communicating objects and classes that are customized to solve a general design problem in a particular context [60]. A pattern has four essential elements: 1) a name, 2) the description of the problem, 3) a solution and 4) the costs and benefits of applying the pattern.

Software engineers are interested in using design patterns to create a repository of solutions for software development. The concept is similar to the source books used by other engineering disciplines, such as civil or electrical engineering. When faced with a particular design problem, civil engineers can consult a catalog of possible techniques. These solutions are based on known successful methods from the industry. The problem with this approach for the software engineering discipline is that SE involves analyzing abstractions that are not based on physical systems. Software design patterns are normative at best, describing solutions given enough similarities in the problem space. In an article published in *Communications of the ACM (CACM)*, Schmidt describe several motivating factors that make it worthwhile to identify these patterns [45]:

1. Success is more important than novelty.
2. Clarity of communication should be emphasized.
3. The need for qualitatively validating of concrete solutions.
4. Good design patterns arise from practical experience.
5. The need to incorporate human factors into software development.

The benefits of design patterns are constrained by the difficulty in applying them in real-world situations. Many design patterns are unnecessarily difficult for the average designer to learn. There is also the problem of making the pattern classifications useful to the practitioner. Cline, a software designer and researcher, argues that some of the classifications do not appear to map to the mental models used by the average developer [32]. Using a scenario-based technique, particular one based on narrative, enables the patterns to better fit into the terminology and knowledge base of the developer.

4.3.1 Overview of Design Patterns

An architect, Christopher Alexander, initially developed the concept of a pattern language [2]. Alexander wished to capture the recurring aesthetic features and functionality of a living space as well as detail standard design solutions. These architectural patterns were created as an aid to the participatory design process, involving user requirements as well as technical guidelines [40].

Software design patterns and frameworks were described to support reusable architecture and detailed design, respectively. A *framework* is a set of components that provide a reusable infrastructure for a family of related applications. Pattern descriptions are often independent of implementation details, whereas, frameworks are semi-complete applications that provide domain-specific functionality. *Patterns* are abstract representations of the problem space that use a particular instance of a framework for portions of the solution. By the same token, a framework may contain several different patterns within its implementation. This discussion incorporates scenario information within patterns. This allows the flexibility inherent in scenario analysis to take advantage of the abstract nature of software patterns. However, scenarios can be associated with frameworks to describe alternate configurations of components within a domain.

In the software engineering textbook *Design Patterns* [60], Gamma et.al. outline three major categories of patterns: *creational*, *structural*, and *behavioral. Creational* patterns are those that deal with initializing and configuring classes and objects. *Structural* patterns seek to separate interface from implementation issues in design. Finally, *behavioral* patterns deal with interactions and collaborations among collections of classes and objects. Within each category, there are numerous identified patterns derived from recurring design tasks. For example, the behavioral patterns are:

1. Chain of Responsibility - All object requests are routed to the responsible service provider in a system.

2. Command - Requests in the system are treated as objects.

3. Interpreter – There is a language interpreter for a grammar of a system.

4. Iterator – An object accesses aggregate elements sequentially .

5. Mediator – An object coordinates interactions between associate objects.

6. Memento – A System snapshot captures and restores object states.
7. Observer – Dependent objects update when a subject changes state.
8. State – Object behavior depends upon its current state.
9. Strategy – There is an abstraction for the selection from different algorithms.
10. Template Method – An algorithm with steps for usage is supplied by a derived class.
11. Visitor – System operations are applied to a heterogeneous object.

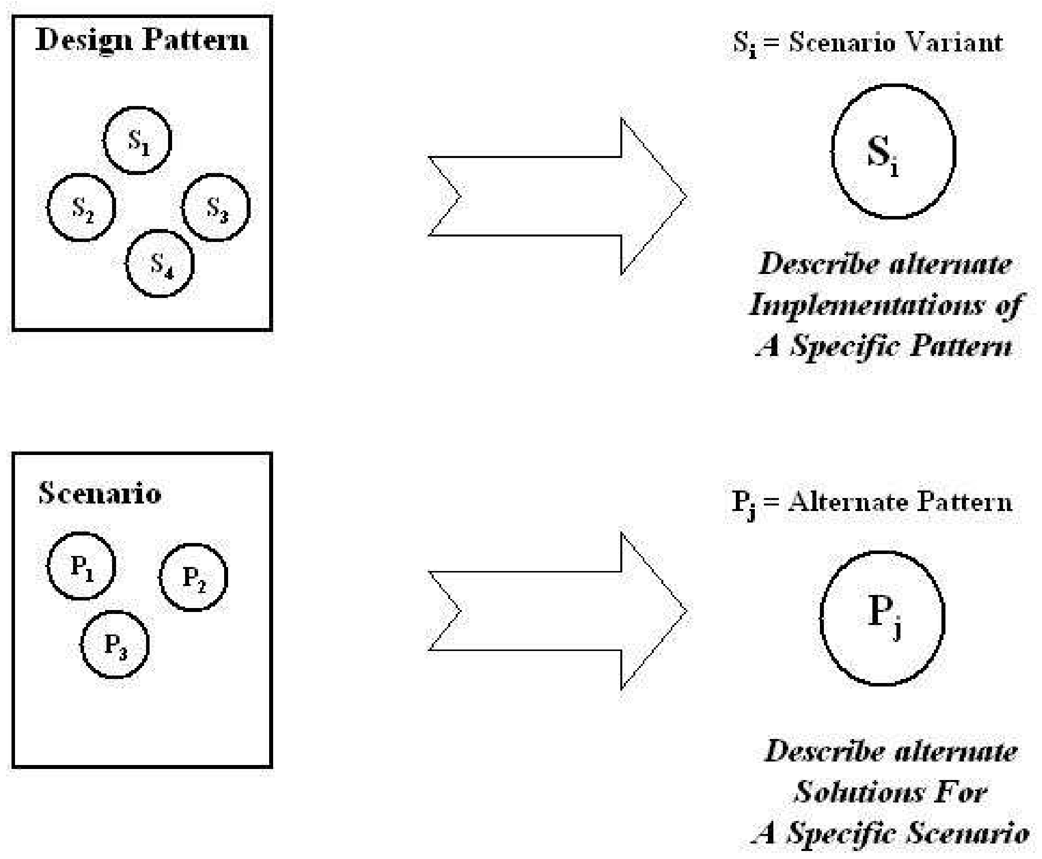
4.3.2 Defining Story Patterns

Figure 38 depicts two approaches for incorporating narrative information with design patterns. In the *Design-Patterns-with-Scenarios* approach, each of the smaller items, S_i, represent scenario variants. A scenario variant is an alternate form of the same scenario. For example, assume we are describing a scenario calling *Getting To The Airport.* One instance of this scenario involves driving through the city and taking the appropriate exit for the airport. A variant on this scenario would be hiring a taxi or shuttle as transportation directly to the airport. With respect to software, each variant within a pattern represents scenario descriptions of alternate implementations of the pattern. These variants could be due to programming language, networking, or even platform differences. The second approach is the *Scenario-with-Design-Patterns* method. In this method, P_j represents several possible solutions for a particular design problem. For example, a scenario can be constructed to represent the requirements of a software system's user interface, based on a particular activity to be accomplished.

4.3.3 Incorporating Scenario Context within Pattern languages

Since the beginnings of discussion on the utility of design patterns, there have been efforts to create languages for representing pattern artifacts. These languages were designed to allow for the direct manipulation of the pattern descriptions in design tools, particularly during requirements analysis. There has also been research into the design of usability pattern languages that could assist the user-centered design process [121].

One example pattern language is *Design Pattern Mark-up Language (DPML)*. This language was developed as part of a research effort to automatically detect design patterns from source code [191]. The researchers viewed design patterns as higher-level abstractions of the object-oriented design within the code. Recognizing these patterns aids program comprehension, code documentation, and validation. DPML was used to create a pattern library, consisting of most of the standard software design patterns as outlined in the descriptions by Gamma et. al. [60]. C++ source code was analyzed to create a class diagram, call graph, and object creation graph. These were then matched against DPML patterns formatted into an XML DOM tree structure. The algorithm was used on four open-source C++ projects. Of twenty-six patterns searched for in over three million lines-of-code, fifteen patterns were detected, with over nine hundred different instances occurring.



Design Pattern
S_1
S_2
S_3
S_4
S_i = Scenario Variant
S_i
Describe alternate Implementations of A Specific Pattern
Scenario
P_1
P_2
P_3
P_j = Alternate Pattern
P_j
Describe alternate Solutions For A Specific Scenario

Figure 38: Story Patterns

Figure 39 is an example of how the narrative structure available in SCML could be used with DPML. This particular situation uses the *Scenario-with-Patterns* approach. The scenario may contain several patterns that represent alternate solutions within the story. The scenario is a description of the process of determining networking requirements for a proposed system. The purpose of the scenario is to examine the risks/benefits of different network implementations. Here, each design pattern is considered to be a character within the scenario. An XML Namespace is used to incorporate DPML elements within the hyperscenario structure. Namespaces are a technique to import information from external languages without conflicting with the grammar of the current language [188]. The `DPML:` prefix identifies those elements that are not part of the SCML grammar. The classes contained in the pattern are described in the element definition. The example code for the *Proxy* pattern in the figure is a modified version from the DPML study [191]. In the network solution scenario of the figure, There would be a character entry for each potential pattern.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE hyperscenario SYSTEM "scml.dtd">
<hyperscenario xmlns:DPML="dpml.dtd"
title="Network Requirement Analysis"
purpose="Examine Alternatives for Implementing System Network"
logline="Risk Assessment of Network Strategies">
<cast>
<character><actor><DPML:DesignPattern='Proxy'>
<DPML:Class id='id10' name='Subject' isAbstract='true'>
<DPML:Operation id='id11' name='Request'
isVirtual='true'><DPML:hasTypeRep ref='id50'/>
</DPML:Operation></ DPML:Class>
<DPML:Class id='id20' name='Proxy'><DPML:Base ref='id10'/>
<DPML:Aggregation ref='id30'/> <DPML:Operation id='id21'
name='Request' isVirtual='true'>
<DPML:defines ref='id11'/> <DPML:calls ref='id31'/>
<DPML:hasTypeRep ref='id50'/></DPML:Operation>
<DPML:Attribute id='id22' name='realSubject'>
<DPML:hasTypeRep ref='id52'/>
</DPML:Attribute</DPML:Class>
<DPML:Class id='id30' name='RealSubject'>
<DPML:Base ref='id10'/>
<DPML:Operation id='id31' name='Request' isVirtual='true'>
<DPML:defines ref='id11'/><DPML:hasTypeRep ref='id50'/>
</DPML:Operation></DPML:Class>
</DPML:DesignPattern></actor></character>
.
.
</cast>
<episode id="000.01" name="Network Service Request">
<goal>Handle External Network Requests</goal>
<scene><setting>Local Area Network</setting>
<event name="Establish Network Connection">
.
.
.
```

Figure 39: Design Pattern Markup Language and SCML

In this example, the episode examines how service requests is handled on the local network by each of the characters (patterns). There are domain-specific actions associated with the patterns in the scenario, some of which could be derived from the operations defined within the pattern. The completed scenario captures design decisions and establish a way of associating non-functional requirements with the system.

4.4 Summary

We developed a proof-of-concept scenario generation tool, Aesop, to highlight the hyperscenario framework and SCML. The prototype is written as a Java applet, using standard web browsers for the user interface. The narrative domain is that of a simple three-dimensional maze game, modified to generate a log file. The objective is to move through the maze, locating colored squares on the floor. The applet captures the user's interaction and uses knowledge of the domain to supply the context. This plot structure is then encoded in SCML. At that point, transformations are applied to the document to create alternate representations of the same story. The usefulness of these story versions is in the extension of the thought experiments that are naturally performed during decision making. Game playing in decision oriented environments, such as logic puzzles and mazes, are inherently pattern matching exercises. Discovering the correct sequence and form of subtasks is accomplished by trial-and-error, until the solution pattern unfolds.

The hyperscenario framework and SCML are submitted as another viable technique for bridging the gap between M & S and C4I systems. In order to share semantically useful data from such a diverse range of systems, it is necessary to describe a conceptual model that can map the rules and behaviors in an abstract way. The use of narrative, in the form of scenarios, as a meta-model makes this representation possible. This narrative structure can be derived from simulation data to generate canonical stories from domain information. SCML leverages XML technologies and can be merged with existing efforts, such as XMSF. The narrative model and the heuristics for story discovery could assist in finding ways to have the simulations inform decision making in C4I systems and for C4I systems to improve the accuracy and utility of simulation tools. This also merges with activities that support the semantic web, specifically DAML+OIL. The scenario ontology contains the concepts, classes, and slot definitions necessary to delineate a narrative domain theory for OIL. DAML as an XML-implementation of an OIL ontology can be mapped to SCML scenario elements, entities, and attributes. This mapping allows SCML to leverage the vast amount of work being done in DAML to support military planning, training, and system development.

Scenario-based design and software design patterns are approaches to handling the complexity and uncertainty inherent in software development. Scenario-based methods are used to capture and track design decisions made during requirements analysis. Design patterns attempt to leverage the best practices in software engineering by recognizing recurring problems and their solutions. This chapter discussed a method that can be used to merge both techniques, creating reusable story patterns that assign further context to proposed design solutions.

Chapter 5

EXPERIMENT DESIGN

It is theory that decides what can be observed.

– Albert Einstein

This chapter describes the experiment done to confirm the hypothesis that a computer-readable scenario format will improve decision-making. To measure an improvement in decision-making, it is necessary to describe the choices, their context, and what constitutes a good decision.

The following sections describe the experimental evaluation of the Hyperscenario Framework as an aid to decision-making. The *Experimental Environment* section describes the maze simulation and the experimental harness. The *Experimental Variables* section describes the variables under consideration during the experiment, as well as their operational definitions. The *Subjects* section states the subjects undertaking the experiment and a brief description of the population from which they are chosen. The *Apparatus and Materials* section describes the experimental instrumentation. The *Test Protocol* section is a description of the plan for executing the experiment and gathering the results. The *Evaluation Method* section discusses how the metrics have been used to analyze the data.

5.1 Experimental Environment

The experiment software is an extension of the Aesop simulation discussed in chapter four. The simulation was modified to add objectives, character interaction, and the ability to capture and process the generated story information. The simulation is an immersive, three-dimensional game representing the point-of-view of someone wandering through a maze. This viewpoint was chosen because it affects how the participants perceive and make decisions about the environment. Navigation through the maze is two-dimensional, involving forward/backward movement and changes in direction. Allowing participants to climb walls, jump over walls, or tunnel through the floor add complexity to the game but does not necessarily add additional insight to their decision-making. If the viewpoint of the maze were two-dimensional, the participant would see the whole maze at once, significantly changing the decision-making aspect.

5.1.1 Aesop Simulation

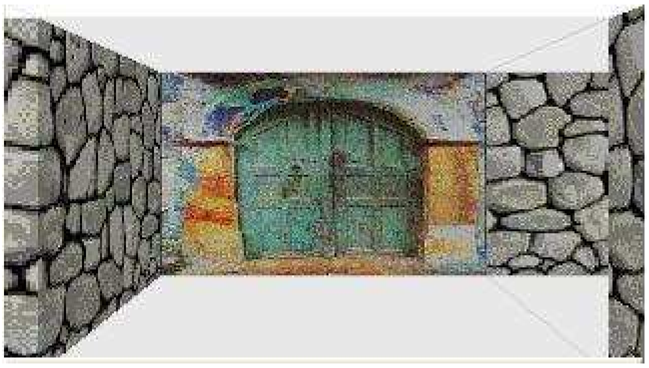
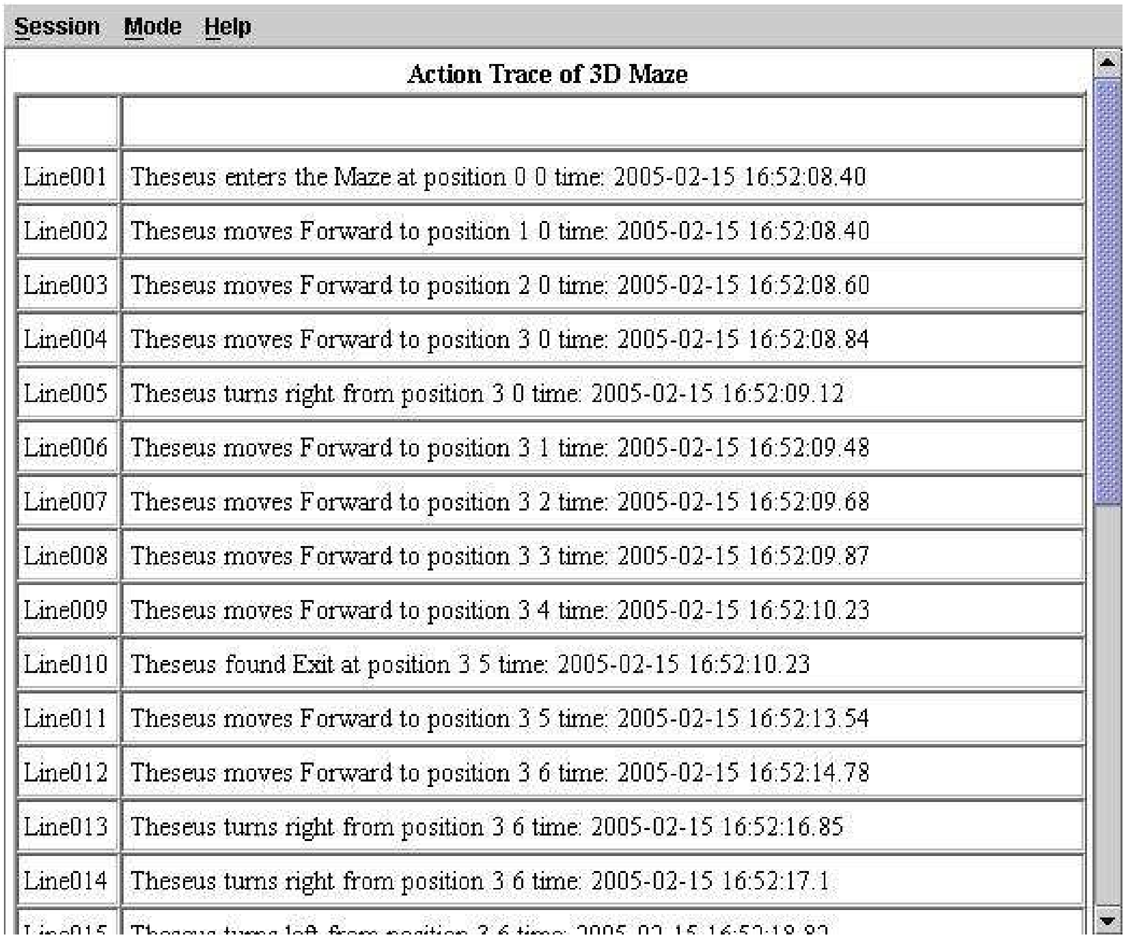


Figure 40: Maze Exit Doorway

The Aesop simulation is a series of three-dimensional images of texture-mapped walls and corridors. The player starts each game session with 500 points, which decreases by twenty points at set intervals during a session, making time a factor. Within the maze there is a closed exit doorway (Figure 40). The player attempts to open the door using the `F2` function key. The maze either opens the exit or supplies the player with a system message. Movement through the maze is controlled using the arrow keys on the keyboard. There are only four movements possible within game: move forward one square, move back one square, turn right, or turn left. There are two avatars in the maze that require player interaction: a dragon and a wizard. Each avatar has a list of items for player selection. Twenty points are assigned to the player's running total for each item selected. Players initiate interaction with a character using the `F1` function key.

A game session begins when the player clicks the start button below the maze viewer. There was a maximum of five minutes of review time between each game session. The game timer reverts to a review timer to count down the seconds remaining. The browser window in the right-hand side of Aesop experiment page contains a view of the previous game session activity (Figure 41). The player uses this browser to plan for the next game session. The player could interrupt the review at any time, starting the next game session. The timer changes automatically from review to game mode. The player could also quit a game session early by pressing the button. If the five minute time expires, the timer automatically changes mode.



Action Trace of 3D Maze

Line001	Theseus enters the Maze at position 0 0 time: 2005-02-15 16:52:08.40
Line002	Theseus moves Forward to position 1 0 time: 2005-02-15 16:52:08.40
Line003	Theseus moves Forward to position 2 0 time: 2005-02-15 16:52:08.60
Line004	Theseus moves Forward to position 3 0 time: 2005-02-15 16:52:08.84
Line005	Theseus turns right from position 3 0 time: 2005-02-15 16:52:09.12
Line006	Theseus moves Forward to position 3 1 time: 2005-02-15 16:52:09.48
Line007	Theseus moves Forward to position 3 2 time: 2005-02-15 16:52:09.68
Line008	Theseus moves Forward to position 3 3 time: 2005-02-15 16:52:09.87
Line009	Theseus moves Forward to position 3 4 time: 2005-02-15 16:52:10.23
Line010	Theseus found Exit at position 3 5 time: 2005-02-15 16:52:10.23
Line011	Theseus moves Forward to position 3 5 time: 2005-02-15 16:52:13.54
Line012	Theseus moves Forward to position 3 6 time: 2005-02-15 16:52:14.78
Line013	Theseus turns right from position 3 6 time: 2005-02-15 16:52:16.85
Line014	Theseus turns right from position 3 6 time: 2005-02-15 16:52:17.1
Line015	Theseus turns left from position 3 6 time: 2005-02-15 16:52:18.82

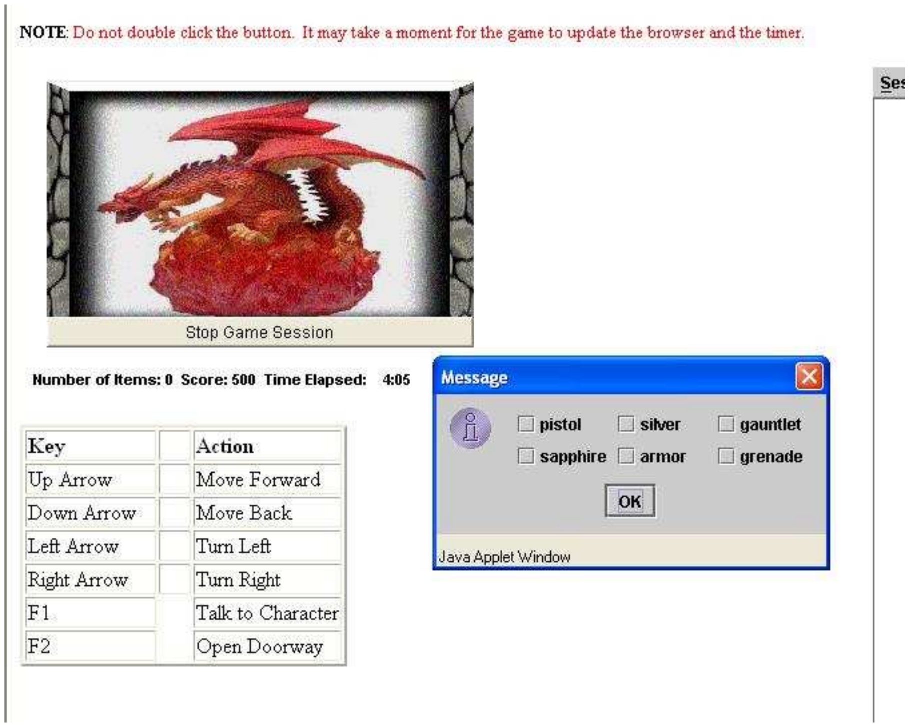
Figure 41: Game Review Browser Window

The main goal of the game is to open the exit doorway. The player must decide which combination of items, avatar interactions, and points will improve the possibility of opening the doorway to successfully complete the game. There are several sub-goals that have to be accomplished to achieve the overall objective: 1) Find a path through the maze, 2) Locate the avatars in the maze, 3) Select the appropriate items from each avatar, 4) Complete the game within five minutes, and 5) Unlock the exit to complete the game.

- *Find a Path through the Maze***:** The maze is a three-dimensional representation of a labyrinth, from the point-of-view of a person walking through the maze. The movement is controlled using the arrow keys on the keyboard.

- *Locate the avatars in the maze***:** The participants are told that there are two avatars to locate with the maze, each one with a list of items to be obtained. The avatars in the simulation are represented by still images on specific walls in the maze.

- *Choose the appropriate items***:** Once a player has located an avatar, the appropriate function key on the keyboard can be used to start the interaction. A dialog box is displayed for interaction with the avatar (Figure 42).



NOTE: Do not double click the button. It may take a moment for the game to update the browser and the timer.

Number of Items: 0 Score: 500 Time Elapsed: 4:05

Key		Action
Up Arrow		Move Forward
Down Arrow		Move Back
Left Arrow		Turn Left
Right Arrow		Turn Right
F1		Talk to Character
F2		Open Doorway

Figure 42: Interaction with Dragon Avatar in the Maze

Each item selected is either a weapon, defensive object, or treasure. These categories are implicit; they are not supplied to the player. The player must determine the categorization and which object combination will open the doorway.

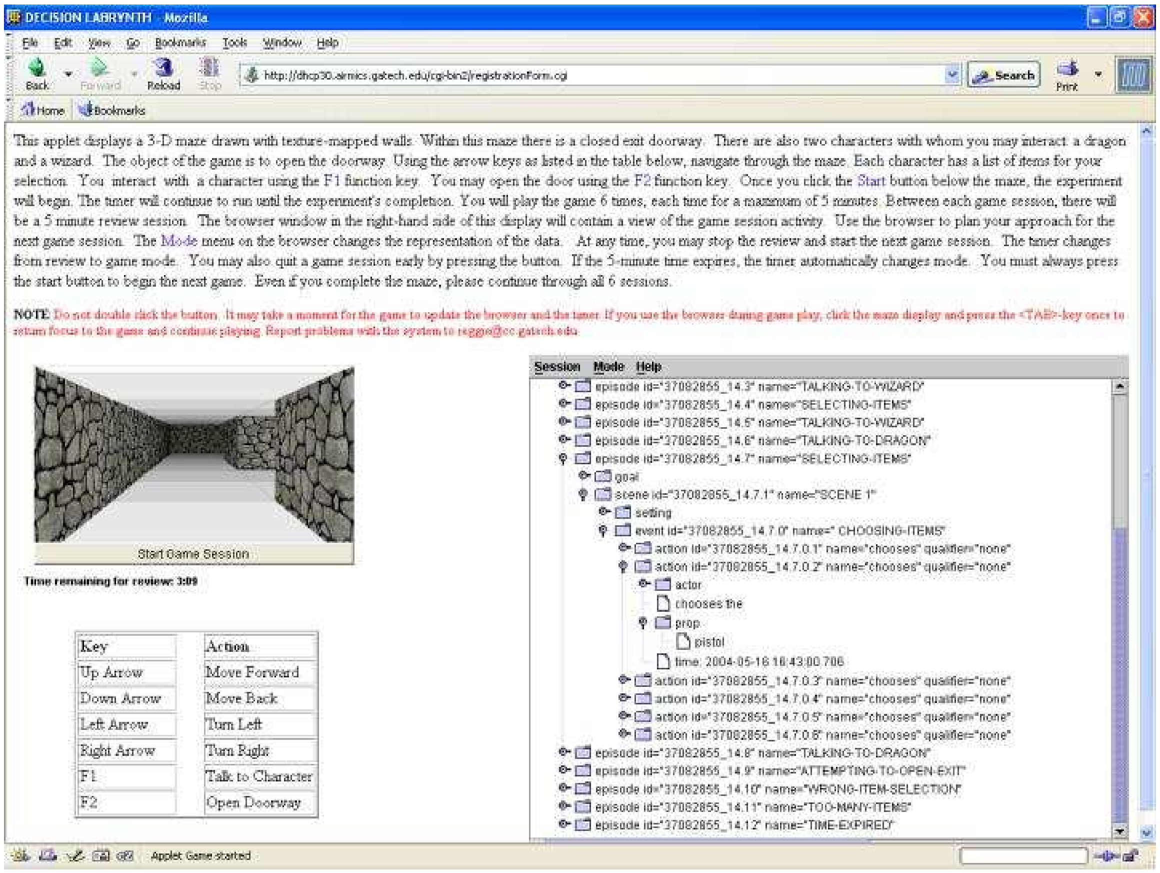
- *Complete the game within five minutes*: Once a game session begins, an embedded clock applet representing a timer displays the current elapsed time. At the end of five minutes the game session will end. If a player opens the exit before that time, the game will end.
- *Unlock the exit to complete the game*: The game exit appears as a texture-mapped image of a closed door. The player cannot end the game and open the door unless the appropriate level of points has been achieved, both avatars have been found, and the correct items have been selected.

5.1.2 Experimental Harness

This same technique was used to determine the maze layout. There were three differing maze layouts in the experiment. This was done to decrease the likelihood of influencing the results by the sharing of information by participants. Three maze layouts were sufficient to decrease the possibility of an exchange of information between players; With six possible arrangements of experiment group-to-layout, there was only a 16.7% chance that a participant would randomly encounter someone with the exact same experiment set-up. The selection of the layouts was implicit; participants were unaware that there was more that one maze layout.

5.1.2.1 User Interface

The user interface for the Aesop web client consisted of three Java applets: 1) maze game, 2) timer, and 3) logfile browser. All three applets were embedded in the web page and visible at the same time. Figure 43 is a snapshot of the GUI as it appears in the Mozilla browser. The top of the web page contains instructions for playing the game. The maze game applets uses texture mapped images to represents walls. The timer below the game window was coordinated with the both the browser and the game. During game play, the timer shows the time used, point total, and current number of items. Between game sessions the timer counts down the time remaining to review the browser. The legend that describes the keystrokes for game navigation and interaction is below the timer.


DECISION LABRYNTH - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://dhcp30.airnics.gatech.edu/cgi-bin2/registrationForm.cgi
Search
Print
Home Bookmarks

This applet displays a 3-D maze drawn with texture-mapped walls. Within this maze there is a closed exit doorway. There are also two characters with whom you may interact: a dragon and a wizard. The object of the game is to open the doorway. Using the arrow keys as listed in the table below, navigate through the maze. Each character has a list of items for your selection. You interact with a character using the F1 function key. You may open the door using the F2 function key. Once you click the Start button below the maze, the experiment will begin. The timer will continue to run until the experiment's completion. You will play the game 6 times, each time for a maximum of 5 minutes. Between each game session, there will be a 5 minute review session. The browser window in the right-hand side of this display will contain a view of the game session activity. Use the browser to plan your approach for the next game session. The Mode menu on the browser changes the representation of the data. At any time, you may stop the review and start the next game session. The timer changes from review to game mode. You may also quit a game session early by pressing the button. If the 5-minute time expires, the timer automatically changes mode. You must always press the start button to begin the next game. Even if you complete the maze, please continue through all 6 sessions.

NOTE: Do not double click the button. It may take a moment for the game to update the browser and the timer. If you use the browser during game play, click the maze display and press the <TAB>-key once to return focus to the game and continue playing. Report problems with the system to reggie@cc.gatech.edu

Time remaining for review: 3:09

Key	Action
Up Arrow	Move Forward
Down Arrow	Move Back
Left Arrow	Turn Left
Right Arrow	Turn Right
F1	Talk to Character
F2	Open Doorway

```
episode id="37082855_14.3" name="TALKING-TO-WIZARD"
episode id="37082855_14.4" name="SELECTING-ITEMS"
episode id="37082855_14.5" name="TALKING-TO-WIZARD"
episode id="37082855_14.6" name="TALKING-TO-DRAGON"
episode id="37082855_14.7" name="SELECTING-ITEMS"
  goal
  scene id="37082855_14.7.1" name="SCENE 1"
    setting
    event id="37082855_14.7.0" name=" CHOOSING-ITEMS"
      action id="37082855_14.7.0.1" name="chooses" qualifier="none"
      action id="37082855_14.7.0.2" name="chooses" qualifier="none"
        actor
        chooses the
        prop
          pistol
        time: 2004-05-16 16:43:00.706
      action id="37082855_14.7.0.3" name="chooses" qualifier="none"
      action id="37082855_14.7.0.4" name="chooses" qualifier="none"
      action id="37082855_14.7.0.5" name="chooses" qualifier="none"
      action id="37082855_14.7.0.6" name="chooses" qualifier="none"
episode id="37082855_14.8" name="TALKING-TO-DRAGON"
episode id="37082855_14.9" name="ATTEMPTING-TO-OPEN-EXIT"
episode id="37082855_14.10" name="WRONG-ITEM-SELECTION"
episode id="37082855_14.11" name="TOO-MANY-ITEMS"
episode id="37082855_14.12" name="TIME-EXPIRED"
```

Applet Game started

Figure 43: Game User Interface

The browser has three pull-down menus. The Session menu allows the participant to review all game sessions available between iterations. The Mode menu is primarily for the narrative group; it supports switching from tabular summary format to a hierarchical view of the story-structured data. The extra menu options are grayed out and non-selectable for the control group. The Help menu supplies group specific information on the use of the game.

5.1.2.2 Data Storage and Retrieval

The data from game sessions and participant demographics were stored in a Microsoft Access database. The database was normalized to prevent redundancy and consisted of six primary tables (Figure 44 on page 129). Four of the tables keep track of session information; two track user information.

1. SessionMetrics Table - Store game session information; primary key `SessionID`.
2. Demographics Table - Keeps track of participant information; primary key `Identifier`.
3. Scenario Table - Contains scenario data; primary key `SessionID`.
4. Good Decisions - Contains counts of good decisions by session;primary key `SessionID`.
5. Bad Decisions - Contains counts of bad decisions by session; primary key `SessionID`.
6. Survey - Contains results of survey taken by participants at end of experiment; primary key `Email` address.

The lines between the tables in figure 44 on the next page depict how tables are joined to access the data. Session tables were joined using the unique `SessionID` key generated by the system. The user data was tracked by an `Identifier` derived from the email address. Most of the data was written directly to the database using Perl scripts with embedded SQL statements. *Java Database Connectivity (JDBC)* calls were embedded in the browser and game applets to retrieve the data for display.

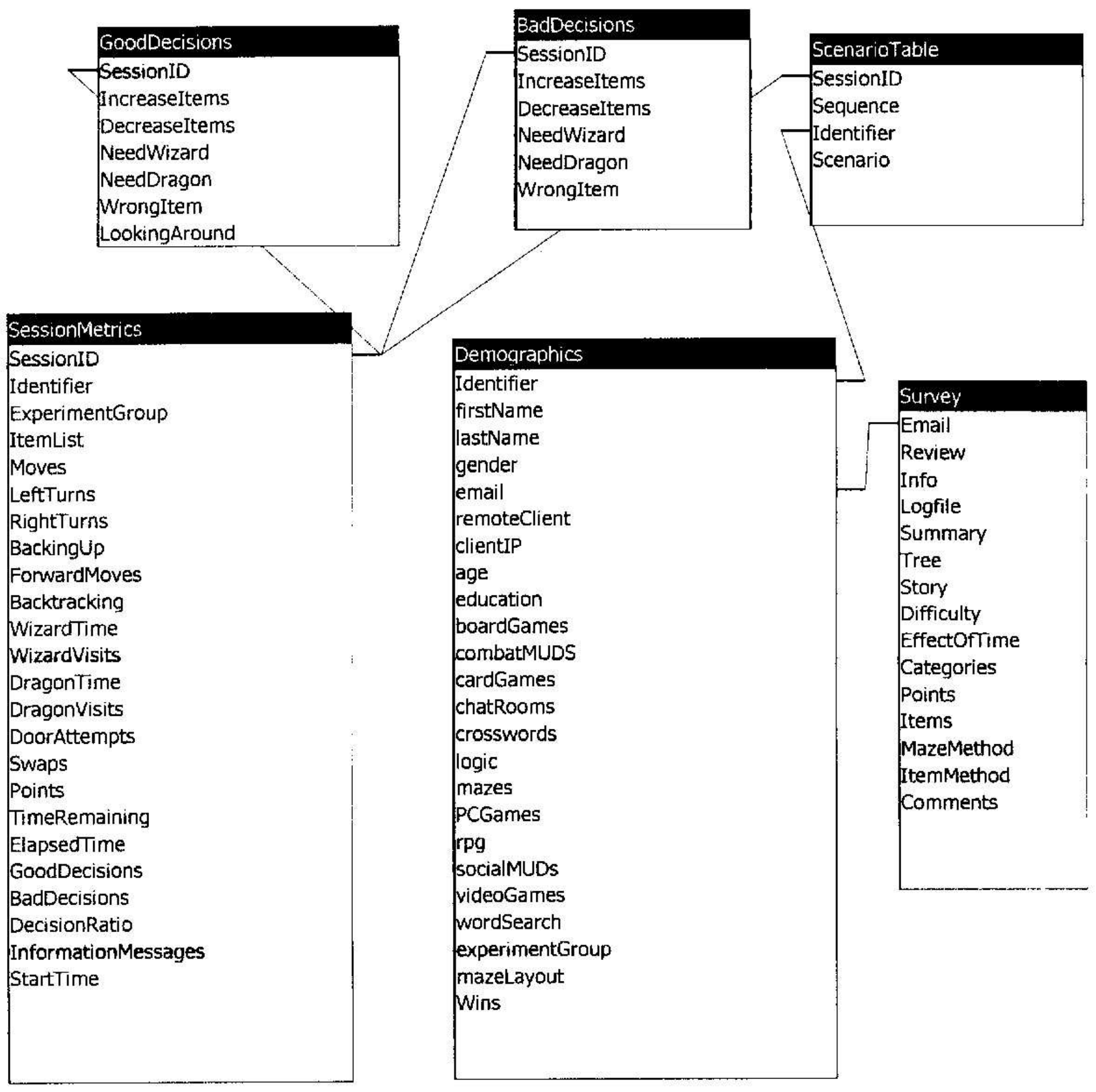


Figure 44: Database Relationships

There were also Perl *Database Interface (DBI)* calls embedded in the CGI scripts for storing the scenario information. Perl DBI is a package that contains standard SQL functions for interacting with relational database [173]. Figure 45 is an example of Perl code to store scenario information into the database. Lines 2-9 initialize the database access information. Lines 11 and 12 open a database connection for queries and updates. The SQL statement to be processed is put together at lines 14-17. The statement includes fields of data to be store in the `ScenarioTable` . Lines 18-20 prepares the SQL statement handle and executes the insert statement against the open database connection.

```
1 # setup the database information
2  $ENV{'DBI_DSN'}="dbi:ODBC:DecisionMaze";
3  $ENV{'DBI_PASS'}="reggie";
4  $ENV{'DBI_USER'}="reggie";
5  $database_name = "DecisionMaze";
6  $db_user = "reggie";
7  $db_password = "xxxxxxx";
8  $databaseString = "dbi:ODBC:$database_name";
9  $db_result;
10 # Connect to the Database
11  $dbh = DBI->connect($databaseString, $db_user, $db_password)
12    or die "Can't connect to Database: $DBI::errstr\n";
13 # Set up insert statement
14  $sql_statement = "INSERT INTO ScenarioTable (".
15     "SessionID, Sequence, Identifier, Scenario) VALUES ('".
16     "$sessionID', $sequence, '$identifier', '$scenario')\;";
17 # execute SQL statement
18  $sth = $dbh->prepare($sql_statement);
19  $sth->execute()
20      or die "Can't execute SQL statement : $dbh->errstr\n";
```

Figure 45: Perl DBI database script

5.1.2.3 Scenario generation

The experiment was designed using the scenario generation components as depicted in Figure 24 on page 88. The maze simulation supplied the event trace that served as input for the scenario generator. The output of the generator was either a logfile or SCML-encoded documents.

The main Perl script responsible for the story generation was called *decisionMaze.cgi*. This was a wrapper script that called each of the generation components and processed the output. The Aesop browser calls *decisionMaze.cgi*, passing it the log file in an input stream. The story generation was accomplished by pattern matching and some additional logic located in the Perl scripts.

5.1.2.4 Filter

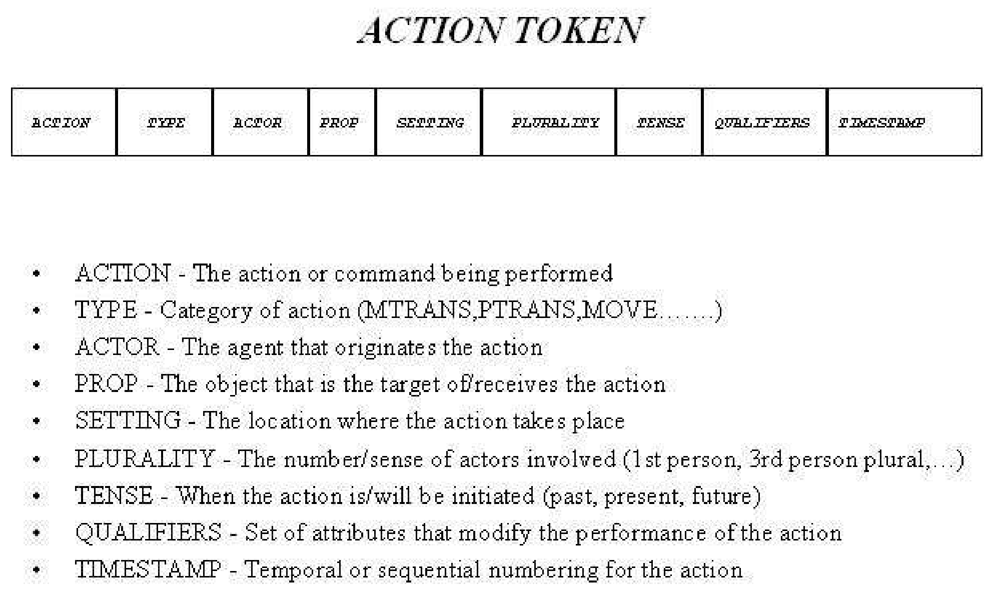
The purpose of the filter script was to remove any background information located in the log file. This component is necessary in a story generator to remove header information that is not relevant to the scenario. The script searches the valid commands or actions and removes lines not containing keywords. The output of the filter is a sanitized log file submitted for processing to the parser script. Figure 46 on page 132 is a portion of the filter script. In lines 6-10, the command list containing the valid actions for Aesop are read into an array. Each line of the original input file is analyzed one line at a time, comparing it against the list of valid actions (lines 17-23). If a line contains an appropriate action, it is written out to the output file.

5.1.2.5 Parser

The parser processes the modified log file and creates a list action tokens. An action token is made up of fields such as: the name of the action, the actor, the qualifiers, time stamp, and any props. (See figure 47). All the fields are not necessary to complete the action token; the minimum information would be an actor-action pair. The parser uses an action list and the associated action categories, as well as a list of valid props. (Table 10). The action list was derived from the actions for maze navigation and player interaction with the avatars in the system.

```


#@cmdlist = <CMDS>; # Build an array from the command file
#chomp(@cmdlist);

for ($i = 0; $i <= $#cmdlist; $i++) {
# print "Command $i before split: $cmdlist[$i]\n";
   ($action) = split(/\|/, $cmdlist[$i]);
   $cmdlist[$i] = $action;
# print "Command $i after split: $cmdlist[$i]\n";
}

# Compare each line of the logfile against the list
# of valid commands. Write the valid commands into
# a new logfile

print "Writing valid commands to log file\n";
while (<OLD>) {
   chomp; # remove end-of-line character
   $line = $_;
   if(isValid($line,@cmdlist)) {
     print NEW ("$line\n");
}
}
print "Removed end-of-line characters\n";
# Close files

#close(CMDS);
close(NEW);
close(OLD);
```

Figure 46: Excerpt of Filter CGI Script



ACTION TOKEN

ACTION	TYPE	ACTOR	PROP	SETTING	PLURALITY	TENSE	QUALIFIERS	TIMESTAMP

- ACTION - The action or command being performed
- TYPE - Category of action (MTRANS,PTRANS,MOVE.......)
- ACTOR - The agent that originates the action
- PROP - The object that is the target of/receives the action
- SETTING - The location where the action takes place
- PLURALITY - The number/sense of actors involved (1st person, 3rd person plural,...)
- TENSE - When the action is/will be initiated (past, present, future)
- QUALIFIERS - Set of attributes that modify the performance of the action
- TIMESTAMP - Temporal or sequential numbering for the action

Figure 47: Action Token

The action categories were taken from work by Roger Schank [161]. These categories were chosen as an example only and are not necessary to create action tokens. Appendix E of this document contains a definition for each of the action categories. The prop categories are not used during story processing. They are implicit categories used to determine successful game completion. It is the choice of particular categories of props that helps achieve the main game objective.

Table 10: Action and Prop Lists

Action	**Qualifiers**	**Category**	**Prop**	**Category**
enters	none	ptrans	pistol	weapon
turns	left,right	ptrans	grenade	weapon
moves	forward, back	ptrans	sword	weapon
found	<character>	grasp	spear	weapon
collides	none	propel	silver	treasure
says	none	mtrans	sapphire	treasure
chooses	<prop>	grasp	diamonds	treasure
discards	<prop>	grasp	gold	treasure
opens	doorway	grasp	gauntlet	defensive
attempts	doorway	grasp	helmet	defensive
exits	none	ptrans	shield	defensive
stops	none	ptrans	armor	defensive

Figure 48 is a portion of the parser Perl script. Each line of the logfile is examined, using the lists and positional information to search for fields. The parser takes the input `$line` variable from the modified logfile that was generated by the filter (Line 3). If an action was a movement (`moves` or `turns`) or the `found` action (Lines 10-20), the parser script looks for qualifiers and positional coordinates. All other commands have no qualifier; those fields were left blank or contain the word `none`. The script determines if there is a valid prop on the line by comparing a substring against an internal list (Lines 24-26). The located fields are joined together into an action token using a pound sign as a delimiter (Line 33). The parser then writes the tokenized version of the log file (Line 34).

```
while (<OLD>) {
   chomp;
   $line=$_;
   ($lineNum,$CONTENTS)=split(/\|/,$line);
   $ACTION=findAction($line,%cmds);
   $actionPosition = index($line,$ACTION);
   $qualifierPosition = $actionPosition+length($ACTION)+1;
   $timePos = index($line,"time: ")+1;
   $location = rindex($line,"position ")+length("position ");
    if($ACTION eq "moves" || $ACTION eq "turns"|| $ACTION eq "found"){
        $QUALIFIER =substr($line,$qualifierPosition,
           index($line,"",$qualifierPosition+1)-$qualifierPosition);
           @coordinates = split(" ",substr($line,$location,$timePos-
           $location));
        $SETTING =
           "XPOS="."$coordinates[0]".","."YPOS="."$coordinates[1]";
          $savedCoordinates = $SETTING;
         } else {
         $QUALIFIER = "none";
         $SETTING = $savedCoordinates;}
      $TYPE=$cmds{$ACTION};
      $actorPos = index($line,"|")+1;
      $ACTOR=substr($line,$actorPos,$actionPosition-$actorPos-1);
      $PROP=findProp($line,@proplist);
      if($PROP eq "0") $PROP="none";
      $propPosition = index($line,$PROP);
      $PLURALITY="singular";
      $TENSE="present";
      $TIMESTAMP=substr($line,$timePos+5);
      @fields=
  ($ACTION,$TYPE,$ACTOR,$PROP,$SETTING,$PLURALITY,$TENSE,$QUALIFIER,
          $TIMESTAMP,$CONTENTS);
      $token=join("#",@fields);
      print NEW "$token\n";}
```

Figure 48: Parser Script

5.1.2.6 Semantic Analysis

The semantic analysis script is responsible for determining event patterns. The analysis tool receives as input the tokenized list of actions from the parser script, still in chronological order. There are twenty-five possible events to be found (Table 11 on page 139). Analysis begins at the top of an action token list, processing each token one at time. Each action token is placed on a stack, while the script attempts to match the action sequence to known events. The fields are separated during processing, so that the actor name and any qualifiers can be used to narrow down the appropriate event. Rules for event changes are processed to empty the stack and create an event pragma. An *event pragma* is a hashtable entry containing event information and the sequence of action tokens detected. The event information includes the name, sequence number, and description of the event. The analysis script creates pragmas from the discovered events and writes them to an output file for processing by the inference engine. The excerpt of the analyzer script in Figure 49 contains code for determining wall events and found events. The script has a bitmap representation of the maze layout for determining wall distances and orientation (Line 1-2). The variable `$mapChoice` is set to the appropriate bitmap for each of the three possible layouts. The rules that the semantic analysis uses for assigning lists of action token to meaningful events include:

- *Rule 1*: The `START-THE-MAZE` event always consists of the first action. The assumption is that the first action is the start of the maze game, therefore, the second action is the beginning of the next event.
- *Rule 2*: A change of direction automatically signals a new event.
- *Rule 3*: If the action is movement, either forward or back, the system stores it on the stack and starts a wall event. If the player is less than four steps in front of a wall, it is considered a `MOVING-AWAY-FROM-WALL` event. If the player is more than four away, it is a `MOVING-TOWARDS-WALL` event (Lines 9-15). In the case were the distance between walls is less than four steps total, it uses a ratio of half the distance to determine the appropriate event (Line 7).

```
$wallDistances =
    calculateDistance($position,$heading,@map."substr($mapChoice,4)");
print "Chose map ".substr($mapChoice,4);
print "\n";
($frontWall,$rearWall)=split(/,/,$wallDistances);
$distanceBetween = $frontWall+$rearWall+1;

$ratioBetweenWalls=$rearWall/$distanceBetween;

if ($action eq "moves"){
   $progress++;
   if($ratioBetweenWalls<=0.5 && $progress<=4){
      $event="MOVING-AWAY-FROM-WALL";
   } else {
      $event="MOVING-TOWARDS-WALL";
   }}
if($action eq "found"){
   $event =
   calculateFoundEvent($qualifier);
} elsif ($action eq "collides") {
   $event="COLLISION-WITH-WALL";
} elsif ($action eq "turns") {
   $event="CHANGED-DIRECTION";
} elsif ($action eq "enters") {
   $event = "START-THE-MAZE";
} elsif( $action eq "says") {
   $event = calculateConversationEvent($actor);
   if( index($contents,"finished",0) != -1) {
       print ERRS "Found the word finished\n";
       $event = "COMPLETED-ITEM-SELECTION";
}}
```

Figure 49: Semantic Analysis Script

- *Rule 4*: When the action is `found`, the analyzer uses the qualifier to determine who or what was found (Lines 16-18).

- *Rule 5*: The `says` action denotes the beginning of an information message or conversation with an avatar. The analyzer uses the `actor` field and certain key words to process the message and categorize it properly (Lines 25-29).

Table 11: Event List

Event	**Description**
START-THE-MAZE	Player starts the game
MOVING-AWAY-FROM-WALL	Player moving away from wall
MOVING-TOWARDS-WALL	Player moving towards wall
COLLISION-WITH-WALL	Player attempts to move forward when in front of wall
CHANGED-DIRECTION	Player adjusts heading either left or right
COMPLETED-ITEM-SELECTION	Player indicates the completion of item selection
RAN-OUT-OF-TIME	Five minute time limit expired
OPENS-DOOR	Player successfully opens doorway
NOT-ENOUGH-POINTS	Player does not have enough points to open doorway
INAPPROPRIATE-ITEM-SELECTION	Player has one or more inappropriate items
TOO-MANY-ITEMS	Player has too many items
TOO-FEW-ITEMS	Player does not have enough items
CANT-OPEN-DOOR-FROM-HERE	Player attempts to open door from a distance
CANT-SPEAK-FROM-HERE	Player attempts to talk to avatar from a distance
NEED-ITEM-FROM-DRAGON	Player needs at least one item from the dragon
NEED-ITEM-FROM-WIZARD	Player needs at least one item from the wizard
CHOOSING-ITEMS	Player selecting items from avatar
ATTEMPTS-TO-OPEN-DOOR	Player attempts to open doorway
LOCATED-THE-DRAGON	Player has located the dragon
LOCATED-THE-WIZARD	Player has located the wizard
LOCATED-THE-DOORWAY	Player has located the exit doorway
QUITS-MAZE-EARLY	Player stops game before time expired
TALKING-TO-DRAGON	Player interacting with dragon
TALKING-TO-WIZARD	Player interacting with wizard
MAZE-SUPPLIES-INFORMATION	Maze supplies information message

5.1.2.7 Story Engine

The story engine code is responsible for taking the event pragmas and creating episodes and scenarios. Episodes and scenarios are processed with simple pattern matching using regular expression search strings.

Table 12 on page 142 is a list of episodes and their goals. The events are examined in reverse order, attempting to match episode patterns. For example, if the `LOCATED-THE-DRAGON` event is found, all events preceding it are assumed to be part of the `LOOKING-FOR-THE-DRAGON` episode. In the set of episodes under consideration the last event in each pattern is unique, so there is no ambiguity in determining the correct episode. The preceding events are placed on a stack until another episode pattern matches. It is assumed that all events between the pattern matches are part of one episode. Figure 50 is the story engine code for processing a list of events into episodes. Line 1 creates a string containing the pattern of events that were gathered as input from the semantic analysis script. The `goal` of each potential episode is stored in the `@patterns` array, along with the sequence of events that make up the episode. Lines 4-7 separates the `goal` of the episode being sought from its event patterns. The event patterns are concatenated into a `$searchString` at line 8. Lines 10-14 uses the `$searchString` and `$eventString` to find out where in the event pattern the episode occurred. If there is a match, the `goal` of the episode is pushed onto a stack for processing into a scenario (Lines 15-19).

```
$eventString=join("",@listOfEvents);
print "The eventString is $eventString\n";
for($goalCounter=0;$goalCounter<=$#goalList;$goalCounter++){
   ($goal,@patterns) = split(/:/,$goalList[$goalCounter]);
   ($goalName,$purpose) = split(/,/,$goal);
   $goal = $goalName;
   print ERR "Searching for $goal episode Pattern\n\n";
   $searchString=join("",@patterns);
   print "The searchString is $searchString\n";
   if($eventString =~ /$searchString/){
      for($i=0;$i<=$#listOfEvents;$i++){ # locate where the pattern
         $searchPattern = substr($patterns[$#patterns],
              rindex($patterns[$#patterns],"*",)+1);
      if($listOfEvents[$i] =~ $searchPattern){
         push(@unsortedGoalArray,"${i}:${goal}:${purpose}");
         print ERR "${i}:${goal}:${purpose}\n";
         print "Found $goal at $i\n";
      }
   }
}
```

Figure 50: Matching Event Patterns to Episodes

Once an episode is matched, the contents of the stack are emptied and inserted into a hash table, using the episode name as the key. Additional episode information, including goal and description, are inserted into the hash table. This hash entry is pushed onto a *Last In-First Out (LIFO)* array structure, which puts it at the beginning of the array. This way, the order of the episodes is preserved by the array indices. This technique is used until the event list is exhausted.

Table 12: Episodes

Episode	**Goal**
STARTING-THE-MAZE	Begin search through the Maze
LOOKING-FOR-THE-DRAGON	Attempting to find Dragon
LOOKING-FOR-THE-WIZARD	Attempting to find Wizard
LOOKING-FOR-THE-EXIT	Looking for Exit
ATTEMPTING-TO-OPEN-EXIT	Trying to open doorway
TALKING-TO-WIZARD	Get information from the Wizard
TALKING-TO-DRAGON	Get information from the Dragon
SELECTING-ITEMS	Select items to carry through Maze
TIME-EXPIRED	Game terminates because time expired
COMPLETED-THE-MAZE	Player completed the Maze
NEED-AN-ITEM-FROM-DRAGON	Maze shares information with Player
NEED-AN-ITEM-FROM-WIZARD	Maze shares information with Player
NOT-ENOUGH-POINTS	Maze shares information with Player
WRONG-ITEM-SELECTION	Maze shares information with Player
TOO-MANY-ITEMS	Maze shares information with Player
TOO-FEW-ITEMS	Maze shares information with Player
MAZE-INFORMS	Maze shares information with Player (doorway)
MAZE-INFORMS	Maze shares information with Player (conversation)
STOPPED-THE-GAME-EARLY	Player ends game before time expired

The scenario patterns are analyzed using a similar technique. Table 13 is a priority list of scenarios. Even though the scenarios had different descriptions, they all had the same stated goal: "Open the Exit Doorway." The scenarios are searched in order down the list, from the most desirable to least desirable. For example, scenarios 1-8 all involve quitting the game early. Scenarios one and three both end up with the participant having chosen the wrong items. However, it is more important that the game was terminated with wrong items and too few items, which is why scenario one is higher on the priority list and will match the pattern first.

Table 13: Scenarios

	Scenario	Description
0	GAME COMPLETED	The player has successfully opened the doorway
1	QUIT GAME WITH WRONG ITEMS AND TOO FEW	Player gave up on game with wrong items and too few
2	QUIT GAME WITH WRONG ITEMS AND TOO MANY	Player gave up on game with wrong items and too many
3	QUIT GAME WITH WRONG ITEMS	Player gave up on game with wrong items
4	QUIT GAME WITHOUT ENOUGH ITEMS	Player gave up on game with too few
5	QUIT GAME WITH TOO MANY ITEMS	Player gave up on game with too many
6	QUIT GAME WITHOUT AN ITEM FROM THE WIZARD	Player gave up on game without obtaining an item from the Wizard
7	QUIT GAME WITHOUT AN ITEM FROM THE DRAGON	Player gave up on game without obtaining an item from the Dragon
8	QUIT GAME BEFORE TIME EXPIRED	Player gave up on game before time expired
9	TIME EXPIRED WITH WRONG ITEMS AND TOO MANY	Time expired and player has wrong items and too many
10	TIME EXPIRED WITH WRONG ITEMS AND TOO FEW	Time expired and player has wrong items and too few
11	TIME EXPIRED WITH WRONG ITEMS	Time expired and player has wrong items
12	TIME EXPIRED WITH TOO FEW POINTS	Time expired and player did not have enough points
13	TIME EXPIRED WITH TOO FEW ITEMS	Time expired and player has too few items
14	TIME EXPIRED NEEDING AN ITEM FROM WIZARD	Time expired and player did not have an item from the Wizard
15	TIME EXPIRED NEEDING AN ITEM FROM DRAGON	Time expired and player did not have an item from the Dragon
16	TIME EXPIRED	Time expired before completing the maze

Figure 51 is the subroutine in the story engine script that is responsible for searching for scenario patterns. Lines 2-3 are the declaration of local variables to be used in the subroutine. The parameters from the main source code are passed to the subroutine and separated into the `$scenario` string and its episode pattern, `@patterns`. The episode list to be searched is generated at Lines 7-10 and converted to the string `$episodeString` at Line 13. The scenario pattern to be matched is changed into a string at Line 12. If the `$scenarioString` occurs anywhere in the episode list, the return value is set to one (true). Otherwise, the scenario was not located, so the return value is set to zero (false)

```
sub findScenario {
   my($scenario,@patterns,@pattern);
   my($scenarioFound,$location,$i);
   ($scenario,@patterns)=@_;
# Build the list of episodes to be searched
   $episodeNumber = @episodes;
   for($episodeSeq=0;$episodeSeq<=$#episodes;$episodeSeq++) {
      for $episode (keys %{ $episodes[$episodeSeq] }) {
          push(@listOfEpisodes,$episode);
      }
   }
   $scenarioString=join("",@patterns);
   $episodeString=join("",@listOfEpisodes);
   chomp($scenarioString);
   chomp($episodeString);
   print "Episode string is ----$episodeString---\n";
   print "Scenario string is ----$scenarioString---\n";
 # Match the patterns one at a time
   if($episodeString =~ /$scenarioString/) {
      $scenarioFound = 1;
   } else {
     $scenarioFound = 0;
   } # End of Pattern Matching
 # Clear the episode array
   while(pop @listOfEpisodes){}

   return ($scenarioFound);
 } # end of subroutine findScenario
```

Figure 51: `findScenario` Subroutine

5.2 Experimental Variables

The most important aspect of evaluating this research was the establishment of a method to measure the relative improvement of the decision-making process. The *independent variable (IV)* was the presence of scenario-structured documentation. The experimental conditions that correspond to the IV were that: 1) the SCML group had SCML-encoded log file information from game sessions 2) the Logfile group did not have access to the SCML encoding. The *dependent variable (DV)* assessed by this experiment was the number of good decisions made by players of the maze game.

<u>*IV Operational Definition:*</u> The scenario-structured documentation consisted of SCML-encoded documents generated by analyzing the log files. The applet browser contains rules and event patterns for the game. Simple natural language processing of the logfiles along with pattern matching of the actions performed were used to construct a plot structure of what occurred during the game sessions.

<u>*DV Operational Definition:*</u> The number of good decisions during game sessions by participants.

The following are categories of metrics gathered during the experiment:

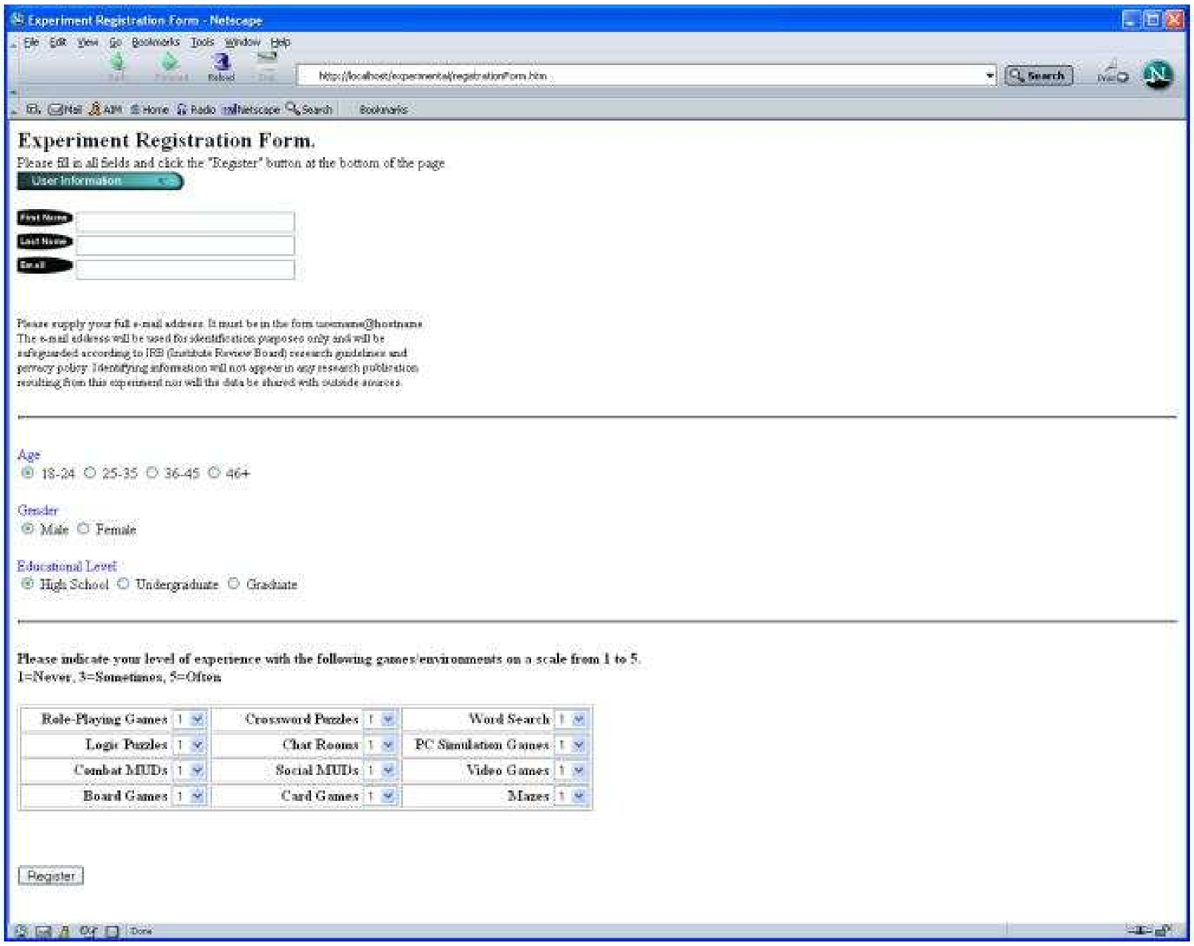
- *Decision Making:* Count of the good decisions per session based on the predefined good decision list. The good decision list is described in the section 5.6.
- *Performance Metrics:* Movement through the maze. Total number of moves, changes in direction, elapsed time, forward movement, backward movement, number of wins.
- *Environment Interactions:* Measure of how often participant interacts with the environment through the avatars or by status messages. Number of informational messages, amount of time per avatar, number of avatar conversations visits.

5.3 Subjects

The subjects for this experiment consisted of volunteers with access to the Internet. The experimental environment includes an Apache [115] web server installed behind a security firewall. The volunteers were solicited through announcements on appropriate newsgroups and e-mail to listserv aliases. The *call for participants (CFP)* form and the research consent form are located in Appendix B. The CFP was not limited to the local university network; the announcement was posted in the SISO listserv, e-mail messages at *Old Dominion University (ODU)* , *Mississippi State University (MSU)* , the *Army Research Laboratory (ARL)*, as well as through regionally-accessible newsgroups on bellsouth.net. Participants registered through an online registration form, which assigns a username and session identification. The registration form also gathers demographic information. In accordance with *Institute Review Board (IRB)* guidelines, children under the age of eighteen were unable to participate in the study.

The sample population used for this experiment were players of the Aesop maze game. Membership in this sample was established by voluntary registration. After reviewing the consent form, the participants were redirected to the registration form as seen in figure 52. The only directly identifiable information solicited from the user was the first name, last name, and e-mail address. The user's first name was assigned as the player name used during game play. The e-mail address was used to generate a unique, ten-digit string for an identifier. This was done by concatenating the first five characters of the e-mail address with the last five digits of the user's IP address.[1] The demographic data gathered consisted of the age, gender, and educational level. The participants who volunteered for the experiment were randomly placed in either the SCML group or the Logfile group. This was accomplished with a simple random number technique that used modulo arithmetic on the system time. Essentially, players were assigned to either group based on whether a calculated value was even or odd. Using this method mitigated potential bias introduced based on participant attributes.

[1]The IP address is a standard part of an HTTP request. Padding with zeros was used when the e-mail address was less than 5 characters.


Experiment Registration Form - Netscape
File Edit View Go Bookmarks Tools Window Help
Reload
http://localhost/experimental/registrationForm.htm
Search
Mail AIM Home Radio Netscape Search Bookmarks
Done

Experiment Registration Form.

Please fill in all fields and click the "Register" button at the bottom of the page

User Information

First Name

Last Name

Email

Please supply your full e-mail address. It must be in the form username@hostname
The e-mail address will be used for identification purposes only and will be
safeguarded according to IRB (Institute Review Board) research guidelines and
privacy policy. Identifying information will not appear in any research publication
resulting from this experiment nor will the data be shared with outside sources.

Age
[✓ 18-24] 25-35 36-45 46+

Gender
[✓ Male] Female

Educational Level
[✓ High School] Undergraduate Graduate

**Please indicate your level of experience with the following games/environments on a scale from 1 to 5.
1=Never, 3=Sometimes, 5=Often**

Role-Playing Games	1	Crossword Puzzles	1	Word Search	1
Logic Puzzles	1	Chat Rooms	1	PC Simulation Games	1
Combat MUDs	1	Social MUDs	1	Video Games	1
Board Games	1	Card Games	1	Mazes	1

Register

Figure 52: Experiment Registration Form

Demographic data was gathered during participant registration via an online form. The participants entered information about their gaming experience using a scale of one to five, with one meaning *never played/used the environment* and five meaning *frequent usage.*

5.4 Apparatus and Materials

Participants in the experiment could use any Internet-connected computer for game play. The maze simulation was accessible through standard web browsers. The software was tested and developed with several web browsers: Netscape 7.0, Microsoft's Internet Explorer 5.0, and Mozilla 4.0 . The software version number and browser acceptability was determined during the registration process. Once a workstation was used for registration, it was assigned as the machine for the game session. The network address and hostname of the machine were captured for this purpose. All documentation, such as game instructions, game log files, and external help documentation was available to both groups. The choice of operating system platform or machine hardware did not affect the data collection.

Aesop was developed on a Pentium 4 3.06GHz laptop machine running Windows XP operating system and running an Apache Web server. The source code was written in Java, PERL scripts, HTML, SQL, and XSL style sheets. The scripts are located on the Web server and transmitted game session information to the client browsers.

5.5 Test Protocol

As the software developer for the Aesop maze game, I am also the subject matter expert for the environment. As discussed in the section on requirements for the hyperscenario framework (3.5.1), SMEs are responsible for describing the stories in a domain. The terminology, naming conventions, meaningful events, and other domain information are captured in the stories that are described by the SMEs. As the SME, I decided what constitutes an action in the Aesop game and what collection of actions are meaningful events. One of the reasons that scenarios are useful in problem solving activities is that they help in knowledge representation. The rules for the domain were written into the source code of the maze game and are reflected in the scenarios that are generated by the experiment.

Two experiment groups, the *Logfile* group and the *SCML* group, are supplied with the same amount of information to begin game play. The display of the data captured for each group between game sessions differed. The difference in representation of the data is declarative versus narrative. A *declarative* view shows a game session as a reproduction of the actions in the maze, just as they occurred. A *narrative* view is a plot-structured view of the data, representing a description of the events and episodes with an educated guess as to what was happening. Each group was presented the same amount of data, just structured into a different form. Both groups used the intra-game session data for planning the next game session.

The experiment consisted of the repetition of a series of maze game playing sessions.

There were six game sessions per participant. Each session was played exactly the same, with the maze simulation reset to its beginning state. The length of individual sessions was limited to five minutes or until the objective is achieved, whichever occurs first. After each session, the participants were given five minutes to review the session to plan the next iteration of the game. The player had the ability to end either review or game play before the time limit had expired. Having six game sessions and a set intra-game review time kept the maximum time to complete the experiment to less than seventy minutes per participant.

5.6 Evaluation Method

Although examining the number of wins by group is a method of gauging improved decision-making, it is not sufficient. It is possible for the decision-making of a group to get better, but not be reflected by the percentage of successful completions. Successfully completing a simulation game with pre-determined objectives is normally done by trial-and-error. Players of video games and PC simulations have to determine patterns in the game that allow them to complete individual tasks. Once these patterns are discovered, achieving the overall goal of winning the game is sped up. These patterns are the rules and policies under which the game's narrative domain operates. The hypothesis that this experiment is attempting to validate is that the hyperscenario framework will improve the decision-making of players in this game simulation. The claim is that the underlying rules and limitations of the game can be discovered much sooner by analyzing a scenario-structured view of the game. For this experiment, improving decision making means that the number of good decisions will increase.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:variable name="leftTurns" select="count(//action[@qualifier=’left’])"/>
<xsl:variable name="rightTurns" select="count(//action[@qualifier=’right’])"/>
<xsl:variable name="forwardMoves" select="count(//action[@qualifier=’Forward’])"/>
<xsl:variable name="backwardMoves" select="count(//action[@qualifier=’Back’])"/>
<xsl:variable name="wizardVisits" select="count(//episode[@name=’TALKING-TO-WIZARD’])"/>
<xsl:variable name="dragonVisits" select="count(//episode[@name=’TALKING-TO-DRAGON’])"/>
<xsl:variable name="doorAttempts" select="count(//episode[@name=’ATTEMPTING-TO-OPEN-EXIT’])"/>
<xsl:variable name="swaps" select="count(//action[@name=’chooses’ or @name=’discards’])"/>
<xsl:template match="/">
<hr/>
Left Turns: <xsl:value-of select="$leftTurns"/>
<hr/>
Right Turns: <xsl:value-of select="$rightTurns"/>
<hr/>
Forward Moves: <xsl:value-of select="$forwardMoves"/>
<hr/>
Backward Moves: <xsl:value-of select="$backwardMoves"/>
<hr/>
Total Moves: <xsl:value-of select="$leftTurns+$rightTurns+$forwardMoves+$backwardMoves"/>
<hr/>
Number of Times visited Wizard : <xsl:value-of select="$wizardVisits"/>
<hr/>
Number of Times visited Dragon : <xsl:value-of select="$dragonVisits"/>
<hr/>
Number of Times attempted Doorway: <xsl:value-of select="$doorAttempts"/>
<hr/>
Number of Swaps: <xsl:value-of select="$swaps"/>
<br/>
</xsl:template>
</xsl:stylesheet>
```

Figure 53: XSL Session Metrics Code

Session metrics were tracked by the client applets as the games were being played. Performance measures (such as number of moves, forward moves, left turns, and right turns) were accumulated in global variables during game navigation. These variables were passed as an array to the Perl scripts for processing. XSL stylesheets made several passes over the scenario documents to calculate variables such as wizard visits, dragon visits, and number of information messages (Figure 53). Lines 4-11 in the figure show how the intrinsic XSL function `count` is used to count the occurrences of variables using X-Pointer syntax for matching. For example, in Line 4 the number of left turns is determined by counting all `action` elements in the SCML document with a `qualifier` attribute set to `left`. Lines 12-32 output a simple HTML-formatted document containing the calculated values separated by horizontal lines.

Table 14: List of Good Decisions

Information Message	**Decision**
Inappropriate Selection	Changing Item Selection
Too Few Items	Increasing Items
Too Many Items	Decreasing Items
Need Item From Dragon	Obtaining Item From Dragon
Need Item From Wizard	Obtaining Item From Wizard
None	Looking Around The Maze

The evaluation of the experiment results will be based on mapping the decision patterns in each session. The most straightforward method for measuring decision making during each event is to have a simple enumeration of possible good decisions. The number of good decisions were calculated for each game session. Table 14 lists the good decisions and their corresponding information messages. Most of the good decisions are intuitive; when the system gives the player a message involving a resource, the player modifies their behavior accordingly. In order to infer that the user made the decision purposefully, there were embedded rules for analyzing the pattern. For example, if the system told the user that they require an item from the dragon, the `TALKING-TO-DRAGON` event has to occur within a set maximum of moves. The only decision that requires no information message was the *Looking Around The Maze* decision. If the player changed direction, then immediately returned to the original heading, that was considered examining the environment. Recall that a turn does not involve movement, but a change of direction. Therefore, a left followed by a right is the equivalent of turning ones head. It is a good decision in the Aesop maze for the player to occasionally stay in one spot and look around to figure out the best path through the maze.

The decision counts were derived by stylesheets that were programmed to detect patterns in the SCML-encoded documents. Figure 54 on page 155 is an excerpt of the XSL stylesheet that was used to count decisions. In this example, the script is counting the number of times the good decision for increasing the number of items selected occurred. For an event to be considered as a purposeful decision, it had to happen within a specific number of events after the original information message. Line 2 shows that the XSL script is analyzing the player's response to the `TOO-FEW-ITEMS` information message from the system. In Lines 4-9, if the `SELECTING-ITEMS` event has occurred as the next event, the script checks to see if the player is performing the `chooses` action. Choosing an item in response to the informational message qualifies as a good decision, so the script writes the string `goodIncreasingDecision` to an output file. Lines 11-31 performs the same processing for `SELECTING-ITEMS` events that are from two to four events away from the informational message. If the user makes a choice of an item more than four events away, it is not considered purposeful and does not count in the good decisions.

```
<!-- Increasing items good decision -->
 <xsl:when test="@name='TOO-FEW-ITEMS'">
   <xsl:choose>
     <xsl:when test="$episodes[$nextNode]/@name='SELECTING-ITEMS'">
      <xsl:if test="$episodes[$nextNode]/scene/event/action/
      @name='chooses'">
goodIncreasingDecision
<br/>
      </xsl:if>
    </xsl:when>
    <xsl:when test="$episodes[$nextNode+1]/@name='SELECTING-ITEMS'">
      <xsl:if test="$episodes[$nextNode+1]/scene/event/action/
      @name='chooses'">
goodIncreasingDecision
<br/>
      </xsl:if>
    </xsl:when>
    <xsl:when test="$episodes[$nextNode+2]/@name='SELECTING-ITEMS'">
      <xsl:if test="$episodes[$nextNode+2]/scene/event/action/
      @name='chooses'">
goodIncreasingDecision
<br/>
      </xsl:if>
    </xsl:when>
    <xsl:when test="$episodes[$nextNode+3]/@name='SELECTING-ITEMS'">
      <xsl:if test="$episodes[$nextNode+3]/scene/event/action/
      @name='chooses'">
goodIncreasingDecision
<br/>
      </xsl:if>
    </xsl:when>
  </xsl:choose>
</xsl:when>
```

Figure 54: XSL Decision Count Code

The data gathered during the experiment was interpreted using the standard statistical techniques available for scientific analysis. Hypothesis testing was done using probability analysis with the appropriately chosen significance level. Descriptive statistics were generated to determine central tendencies of the data and amount of dispersion (variability). These descriptive statistics include (but are not limited too) the arithmetic mean, standard deviation, variance, and z-scores (standardized values). The next chapter presents and analyzes the results from the maze experiment. The data analysis is described in detail in chapter 6.

5.7 Handling Threats to Validity

There are outside factors that can limit the validity of this experiment by introducing variables that affect conclusions. These outside factors can be mitigated by the appropriate experimental design that takes into account these threats to validity. *Threats to validity* refer to the manner in which variables can influence the results of an experimental study and its generalizability to the population under consideration [78]. There are internal and external threats to validity. *Internally validity* occurs when the results on the dependent variable are solely attributed to the independent variable. *External validity* is a measure of how well the study can be generalized to groups beyond the experiment sample.

There is extensive literature about research methods to decrease the affects of outside factors, particularly in the social sciences([118] , [176], and [130]). This is because many psychological and sociological experiments involve human subjects, which account for many of the unwanted variables. The next sections describe the individual factors and how they were handled for this experiment.

5.7.1 Effects to internal validity

1. *History* - *History* refers to any event outside of the research study that can affect performance. Participants performed one trial of six sessions, making it difficult to establish a history. This discouraged the memorization of the maze layout that could occur during multiple trials. The random selection of maze layout and experiment

group also decreased the amount of information that could be shared between participants.

2. *Maturation* - *Maturation* is only a factor if the experiment takes place over a long period of time, so that basically everyone would improve. The experiment site was available for a week to ten days. Participants were instructed not to share information about the game and its solution to other potential volunteers.

3. *Selection of subjects* - The volunteers for this experiment were recruited based only on their ability to access the Internet and locations where the request for participation was posted. Although demographic data was gathered for post-experiment analysis, assignment to experimental groups was randomized.

4. *Instrumentation* - *Instrumentation* refers to consistency in measurement and devices used throughout the experiment. Both groups had access to the same user interface and the metrics collected did not differ by group.

5.7.2 Effects to external validity

1. *Reactive effects* - Reactive effects are effects due to pretesting subjects. The participants were exposed to the game environment only once, during the actual run of the experiment. Anyone who participated in any pilot study or during development was precluded from the actual experiment.

2. *Selection bias vs. Experimental variable* - *Selection bias* of this type occurs when participants are chosen who have a characteristic that is known to affect the experiment variable under consideration. The randomization of group selection decreased any bias due to prior game experience or knowledge of similar maze environments.

3. *Self-Selection bias* - *Self-Selection bias* occurs when there is non-random sampling of membership within a group or category that is hypothesized to affect a variable of interest [98]. Recruitment of volunteers for the experiment was not limited to the local university. Messages were posted on regional newsgroups, distributions to three other universities, e-mail messages to a government site, and listserv posts to a simulation

news group. The assignment to experiment group was random and not based on any user attribute. Also, since the maze domain was created for this experiment, there was no direct characteristic of a participant that makes them more likely to make a good decision in the Aesop maze. The impact of any indirect user characteristic was minimized by the randomization of group selection.

4. *Multiple treatments* - Multiple treatment bias happens when participant are involved in an experiment multiple times. Participants performed the maze experiment once.

5.8 Summary

This experiment was designed to validate the impact of scenario-structured information on decision-making. The subjects for the experiment were volunteers who had access to the web-enabled maze simulation. The only requirement, in terms of apparatus and materials, was the use of a Java-compatible browser, Netscape, Mozilla or Internet Explorer. The source code was an implementation of a scenario generator, with each component written as Perl scripts or Java applets. The simulation supplied the event traces that were analyzed by the system to determine plot structure. The user interface allowed participants to play the game and perform planning for successive iterations. Players were randomly assigned to either the Logfile group, who received direct log file information, or the SCML group that received scenario-structured documents for review. The performance and decision metrics were stored in a database for post-experiment analysis and comparison. Evaluation of the experiment is based on comparing the effect on the dependent variable, the number of good decisions, by the presence (or absence) of the independent variable, scenario-structured information.

Chapter 6

DATA ANALYSIS & RESULTS

It is a capital mistake to theorize before one has data

– Sir Arthur Conan Doyle

This chapter contains an analysis of the results of the empirical study. The experiment was conducted over a two-week period, with over 50 volunteers participating. Scenarios were examined to capture performance metrics and quantitative decision-making data. Each potential threat to validity and the method for mitigating that threat was discussed. The first section discusses the impact of user demographics and experience level on the results. The second section describes performance metrics used to infer player decision making. Section three is an analysis of the descriptive decision statistics for both experimental groups, outlining the central tendencies, spread, and frequency distribution. Finally, hypothesis testing using a z-test comparison of sample means is used to validate the results of hyperscenario-directed decision-making.

6.1 Effect of Experiment Set-up on Validity

Table 15 is the demographic information on experiment participants showing counts and percentages for gender, age, and educational level for both experimental groups. Recall that membership on the groups was randomized without taking into account any of the demographic information. As can be seen from the table, the percentage of participants based on gender were fairly evenly distributed. In the age category, the percentages were also fairly consistent at each level across groups. There is a slightly larger percentage of 36-45 year olds on the SCML group. There are not enough data points to determine if this difference is statistically significant. The age data values for the experiment groups can be compared by using descriptive statistics.

Table 16 contains the statistical information on the age for the Logfile group. The median

Table 15: Demographics

	Logfile	SCML	% Logfile	%SCML
		GENDER		
Male	14	20	58.3%	60.6%
Female	10	13	41.7%	39.4%
		AGE		
18-24	16	22	66.7%	66.7%
25-35	3	4	12.5%	12.1%
36-45	3	5	12.5%	15.2%
46+	2	2	8.3%	6.1%
		EDUCATION LEVEL		
High School	3	1	12.5%	3.0%
Undergraduate	14	25	58.3%	75.8%
Graduate	7	7	29.2%	21.2%

age for the group is 26.6 years. The confidence level, using a level of significance of 0.05, is 3.806. The level of significance establishes a 95% certainty in the results. The confidence interval for the age would be 26.6 ±3.806. This means that there is a 95% certainty that age values in the range from 22.84 years to 30.45 are statistically the same as that of the Logfile group. The descriptive statistics in Table 17 are the age values for the SCML group. The mean age is 26.561. This is within the confidence interval for the Logfile group, so the averages are statistically the same.

Table 16: Logfile Age Statistics

Mean	26.646
Standard Error	1.840
Median	21
Mode	21
Standard Deviation	9.014
Sample Variance	81.250
Kurtosis	0.053
Skewness	1.277
Range	25
Minimum	21
Maximum	46
Sum	639.5
Count	24
Confidence Level(95.0%)	3.806

Table 17: SCML Age Statistics

Mean	26.561
Standard Error	1.525
Median	21
Mode	21
Standard Deviation	8.759
Sample Variance	76.715
Kurtosis	-0.126
Skewness	1.219
Range	25
Minimum	21
Maximum	46
Sum	876.5
Count	33
Confidence Level(95.0%)	3.106

To calculate an average educational value for the groups, each level was given a rating of one, two, or three representing high-school, undergraduate, and graduate levels, respectively. The ratings were then averaged for each group, yielding an average education level of 2.18 for the SCML group and 2.16 for the Logfile group. Tables 18 and 19 are the educational statistics for the groups. Using the confidence level and mean of the SCML group, the confidence interval is from 2.017 to 2.347. The mean for the Logfile education is within this range and is statistically the same as the SCML group educational mean.

Table 18: Logfile Education Statistics

Mean	2.167
Standard Error	0.130
Median	2
Mode	2
Standard Deviation	0.637
Sample Variance	0.406
Kurtosis	-0.368
Skewness	-0.143
Range	2
Minimum	1
Maximum	3
Sum	52
Count	24
Confidence Level(95.0%)	0.269

Table 19: SCML Education Statistics

Mean	2.182
Standard Error	0.081
Median	2
Mode	2
Standard Deviation	0.465
Sample Variance	0.216
Kurtosis	0.834
Skewness	0.674
Range	2
Minimum	1
Maximum	3
Sum	72
Count	33
Confidence Level(95.0%)	0.165

There were seven categories of experience to be assessed. For each type of experience, the user was asked to gauge, on a scale of one-to-five, their level of knowledge or use of the game. A selection of one represented no experience with the activity; three represented some experience with the activity; five represented frequent use of the activity. A participant at level five can be viewed as a person who did the activity as a frequent gamer.

Table 20: Logfile Experience By Maze

	Board **G**ames	**C**ombat **MUDS**	**C**ard **G**ames	**C**hat **R**ooms	**C**rosswords	**L**ogic **P**uzzles	**V**ideo **G**ames
Maze 1	2.40	1.80	2.80	2.20	1.80	2.20	2.60
Maze 2	2.83	1.67	3.17	2.00	2.00	2.83	3.00
Maze 3	4.00	1.08	3.46	1.62	3.08	3.38	3.23

Table 20 is the data on experience for the Logfile group. The three rows correspond to the different maze layouts. This table was constructed to determine if the amount of experience was distributed evenly among the maze layouts. There is a slightly larger average value for board games and crosswords for maze layout three but it does not appear to be enough to have impacted the overall results for the logfile group

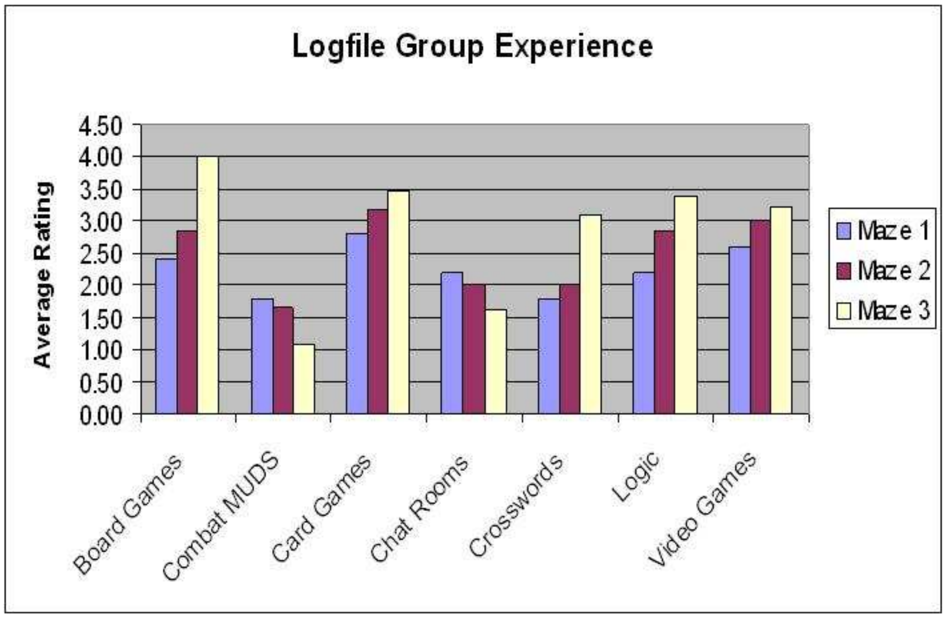


Figure 55: Logfile Experience By Maze

Figure 55 is a graphic depiction of the Logfile experience data, color-coded to represent the maze layouts. It is easier to see the variations in experience using the chart to represent the data. The SCML group experience is distributed evenly across the maze layouts, as shown in Table 21.

Table 21: SCML Experience By Maze

	Board Games	Combat MUDS	Card Games	Chat Rooms	Crosswords	Logic Puzzles	Video Games
Maze 1	3.54	1.85	3.31	2.92	2.54	2.54	3.46
Maze 2	3.80	2.40	3.50	3.10	2.60	2.70	4.10
Maze 3	3.11	1.11	3.11	1.33	2.44	2.78	2.78

Figure 56, the graphic for the SCML group, also shows a fairly consistent level of experience across the categories. When comparing the diagrams for both groups, it appears that no single category of experience was prevalent on either group. Again, there are not enough data points by maze layout to establish if there variations are statistically meaningful. Examining the experience values across all layouts by group yield a better comparison.

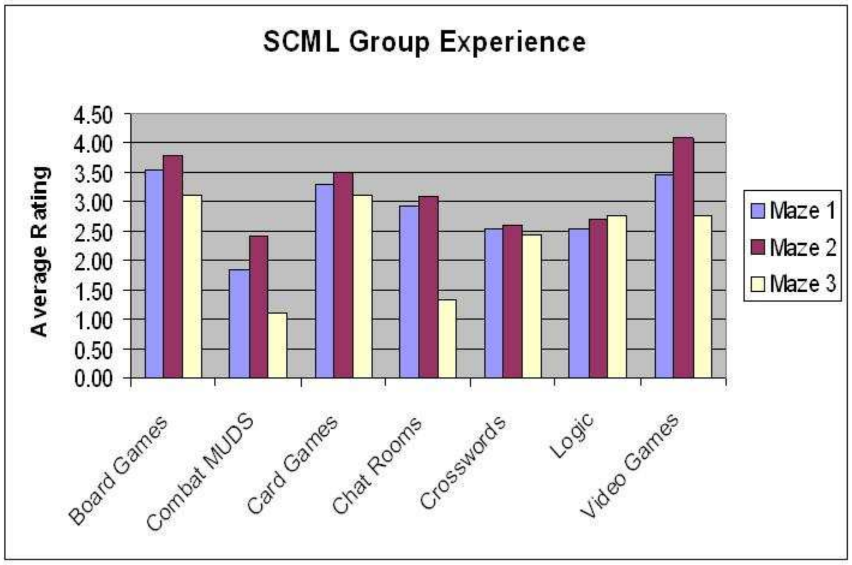


Figure 56: SCML Experience By Maze

The Table 22 lists the average for each category of experience, organized by experiment group.

Table 22: Experience Level By Group

	Board Games	Combat MUDS	Card Games	Chat Rooms	Crosswords	Logic Puzzles	Video Games
Control	3.38	1.38	3.25	1.83	2.54	3.00	3.04
SCML	3.50	1.81	3.31	2.53	2.53	2.66	3.47

Figure 57 is the graphical depiction of the data from the experience table. The board games, card games and crosswords are almost equal, with a negligible variation between the groups. The same descriptive statistics used earlier can be used to determine if there is a statistical significance to the variations in the other experience levels.

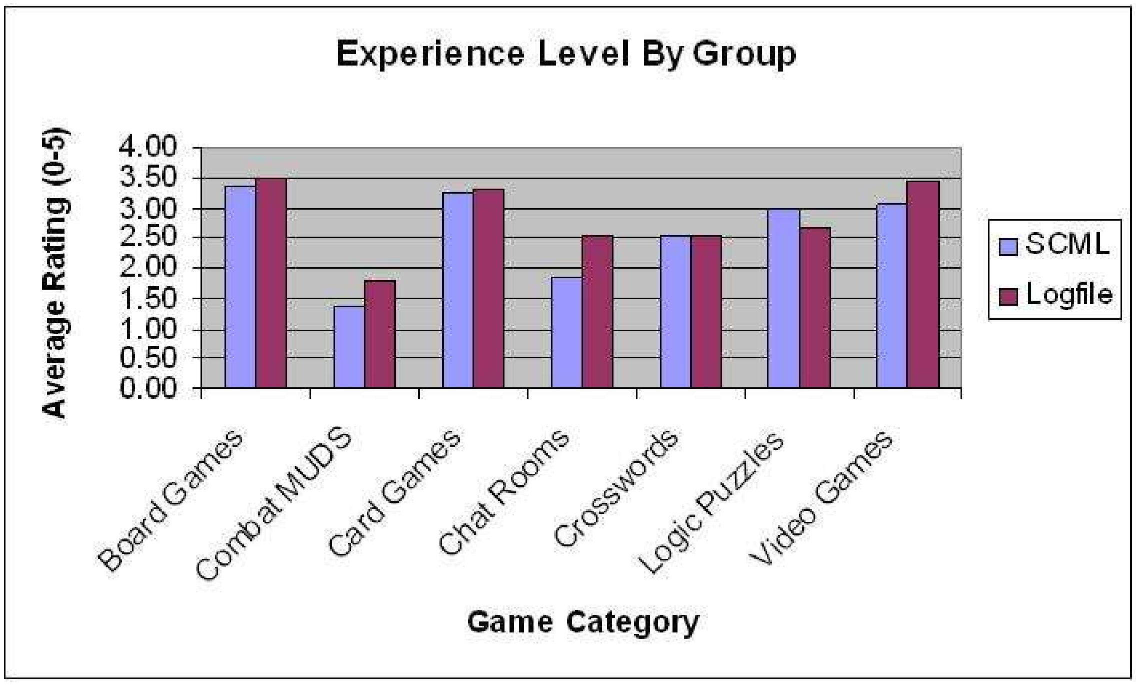


Figure 57: Average Experience By Group

The following tables are descriptive statistics for the combat MUD experience for the experiment groups (Tables 23 and 24). The confidence interval for the SCML MUD experience level is 1.78 ±0.507 which ranges from 1.273 to 2.287. The Logfile group's MUD experience is statistically the same. Appendix E contains similar information for the chat room, logic puzzle, video game, and board game experience levels by group. All of the levels are statistically the same and therefore did not impact the results of the experiment.

Table 23: Logfile MUD Experience Statistics

Mean	1.375
Standard Error	0.145
Median	1
Mode	1
Standard Deviation	0.711
Sample Variance	0.505
Kurtosis	1.368
Skewness	1.671
Range	2
Minimum	1
Maximum	3
Sum	33
Count	24
Confidence Level(95.0%)	0.300

Table 24: SCML MUD Experience Statistics

Mean	1.788
Standard Error	0.249
Median	1
Mode	1
Standard Deviation	1.431
Sample Variance	2.047
Kurtosis	1.561
Skewness	1.760
Range	4
Minimum	1
Maximum	5
Sum	59
Count	33
Confidence Level(95.0%)	0.507

6.2 Using Performance Metrics to Infer Decision-making

There were several performance metrics that were tracked or calculated during game play that could be used to evaluate the decision making patterns of the players. As we have defined it for this experiment, decision making is the process of making choices after consideration. The context for those choices are represented in the domain. Each action, event, or episode in the maze could constitute a decision by the player in order to solve the maze. We can use some performance metric data to infer the strategy being applied by the players in the game and determine decisions that were being made.

Session	**S**CML	**L**ogfile
1	4.6	4.8
2	5.2	6.6
3	9.8	4.6
4	14.5	2.3
5	10.7	3.2
6	0.8	4.3

Table 25: Average Doorway Attempts

The number of attempts to open the doorway is an indicator of the kind of strategy the participant was using to solve the game (Table 25). This table shows the average number of attempts per group, organized by game session. The number of attempts for the SCML group continued to increase through iteration four. At this point, the attempts decreased to the lowest value by the last session.

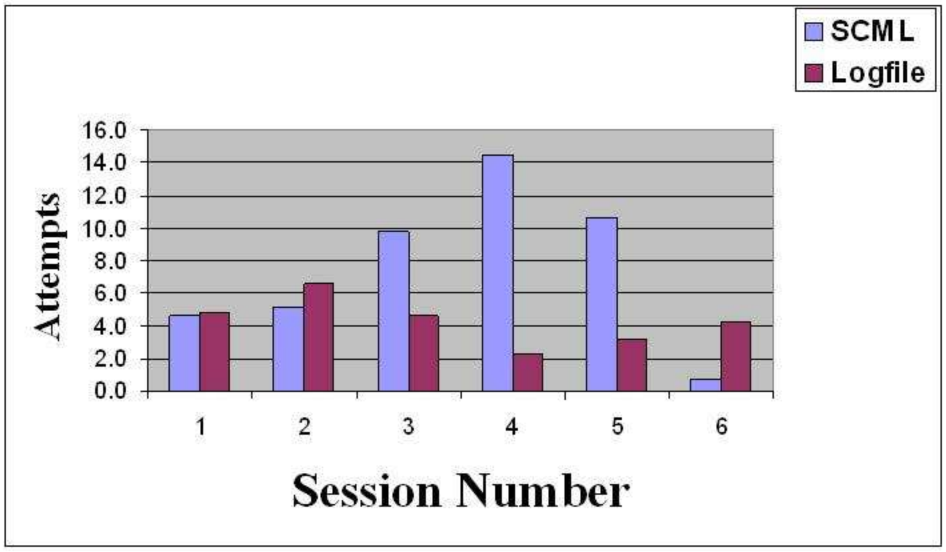


Figure 58: Average Doorway Attempts

Figure 58 makes it easier to compare the trends between the two groups. There is a definite spike in attempts for the SCML group during session four. Overall, the SCML group attempted the doorway more often than the Logfile Group. As the domain expert, I can say that this is the most effective strategy for solution, because it allows the participants to narrow down choices for each system informational message. For example, when the system displays a `TOO-FEW-ITEMS` message, a player can use the strategy of adding items one at a time until that message is no longer shown when the door is tried.

The particular choice of items was one of the most difficult values to determine by the players. Purposefully, the system does not say what makes a choice inappropriate, just that there is at least one inappropriate choice within the users current inventory. Exhausting every possible combination of items would be prohibitive, given the time limits on each session. The player had to determine any implicit characteristics that would aid in the correct selections. Measuring the number of swaps indicates how intensely the player is attempting to find the right sequence of items.

Session	**S**CML	**L**ogfile
1	15.8	15.8
2	18.0	13.4
3	21.0	12.0
4	22.3	5.8
5	17.3	7.4
6	4.8	8.3

Table 26: Average Swaps

Figure 59 shows the trend of average swaps between the groups. Again, there is a definite increase of swaps during the middle sessions for the SCML group. The Logfile group decrease the number of swaps and then begin to increase them.

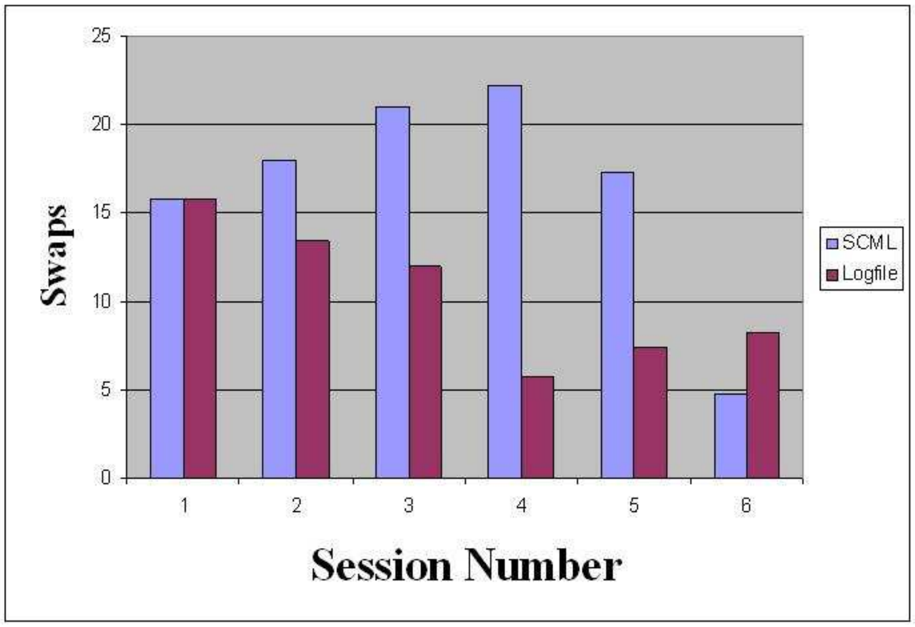


Figure 59: Average Item Swaps

Both the number of attempts and number of swaps had an effect on the number of points. The number of points decreases periodically during game play, so continuing to try the doorway decreases the running total of points. Every time an item is removed from the player's inventory the points also decrease. In Table 27, the average total points for the SCML group is lower for the first four sessions. This might indicate that the SCML players were less concerned about their point totals during the beginning sessions than finding the appropriate sequence of items.

Session	**S**CML	**L**ogfile
1	316.0	388.0
2	344.0	440.0
3	253.3	440.0
4	395.0	395.0
5	413.3	404.0
6	510.0	375.0

Table 27: Average Points

Figure 60 depicts the trend by session. There is a definite dip in total points during the third session. This would make sense, if the participants were rapidly swapping items, trying to determine the right combinations while trying the doorway as often as possible. The SCML group ends the sessions with more total points, ostensibly because they had determined the solutions to the maze.

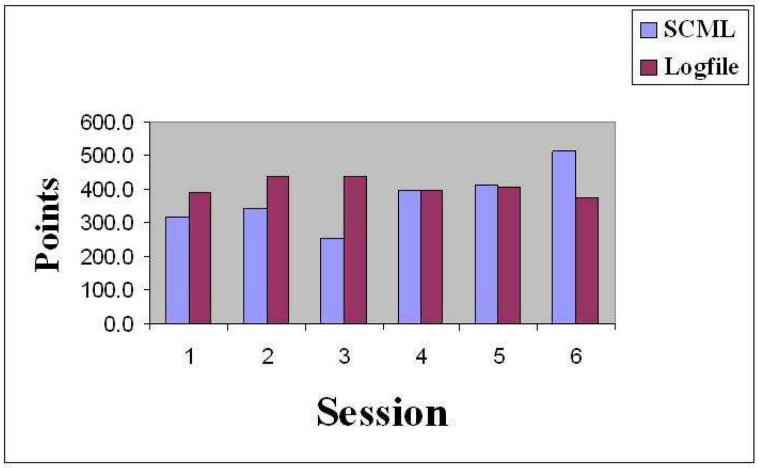


Figure 60: Average Points

6.3 Frequency Distribution and Descriptive Statistics

As described in the chapter on the experimental design, there is an list of good decisions for this maze environment. This decision table was used to establish a count of the number of good decisions made by each group during game play. From this count, statistics describing the central tendency, spread and frequency distribution of decision-making were calculated. Table 29 shows the descriptive statistics for the experiment. Each row in the table represents the data for a session indexed by the unique seven digit session ID. The first column in the table is the session ID followed by the second column which is the actual number of good decisions for that particular session.

The central tendency shows how a set of values converge. The central tendencies are presented by the mean and the median. The average number of good decisions in the Logfile group was 24.6; the median was 26 good decisions. In the SCML group, the mean and median were 34.3 and 24.5, respectively. It is interesting to note that while the medians for the two groups are very similar, the means are much further apart. This is supported by the measures of variability: the standard deviation and the range.

Logfile Good Decisions			
Mean	**S**tdDev	**M**edian	**R**ange
24.64	16.98	26	55

Session ID	**C**ount	**D**ev	**Z**-Score
4724100	0	-24.64	-1.45
2616130	2	-22.64	-1.33
3344410	10	-14.64	-0.86
3304540	49	24.36	1.43
2144250	11	-13.64	-0.80
2329000	35	10.36	0.61
1448130	37	12.36	0.73
3204120	38	13.36	0.79
2952200	6	-18.64	-1.10
2324500	32	7.36	0.43
9203300	9	-15.64	-0.92
3904260	6	-18.64	-1.10
2096480	7	-17.64	-1.04
1708180	55	30.36	1.79
1100230	44	19.36	1.14
3784900	46	21.36	1.26
3524360	4	-20.64	-1.22
3832220	26	1.36	0.08
1588430	26	1.36	0.08
1592150	14	-10.64	-0.63
3268210	12	-12.64	-0.74
3180160	33	8.36	0.49
3268500	36	11.36	0.67
3044340	39	14.36	0.85
3828330	39	14.36	0.85

Table 28: Logfile Decision Statistics

SCML Good Decisions

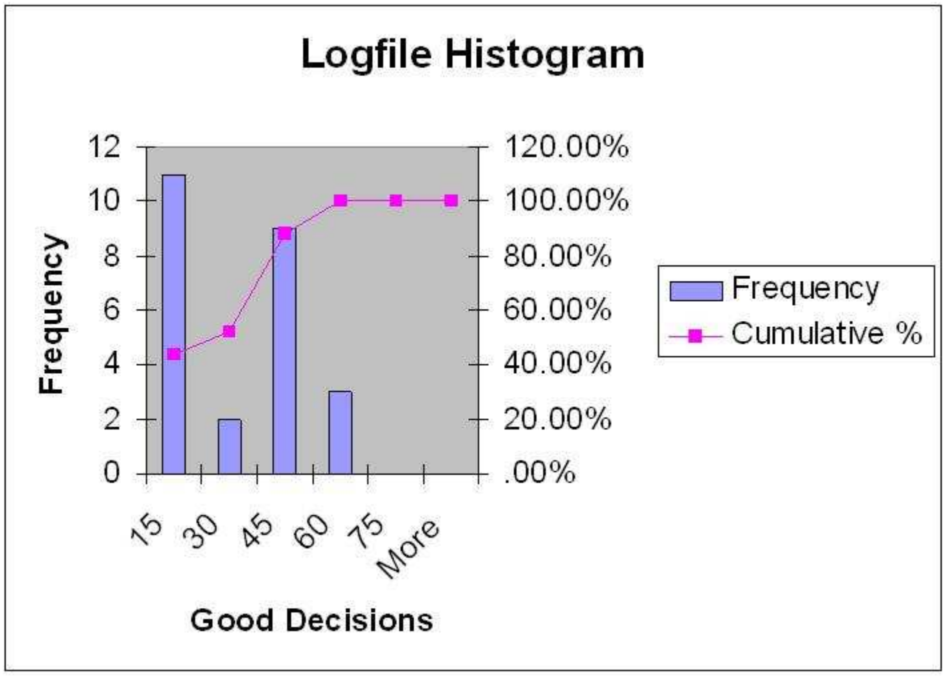
Mean	StdDev	Median	Range
34.375	29.65	24.5	107

Session ID	Count	Dev	Z-Score
1040270	0	-34.375	-1.16
3205000	1	-33.375	-1.13
1744330	2	-32.375	-1.09
1928160	6	-28.375	-0.96
5482400	24	-10.375	-0.35
2944520	31	-3.375	-0.11
3412590	40	5.625	0.19
2924180	48	13.625	0.46
2444140	95	60.625	2.05
1488490	30	-4.375	-0.15
2468370	107	72.625	2.45
3424500	16	-18.375	-0.62
1100240	11	-23.375	-0.79
2776550	62	27.625	0.93
2128310	14	-20.375	-0.69
3144250	10	-24.375	-0.82
4036450	11	-23.375	-0.79
2272570	22	-12.375	-0.42
1168380	23	-11.375	-0.38
2988580	64	29.625	1.00
1484300	55	20.625	0.70
5361700	56	21.625	0.73
2984520	72	37.625	1.27
1088320	25	-9.375	-0.32

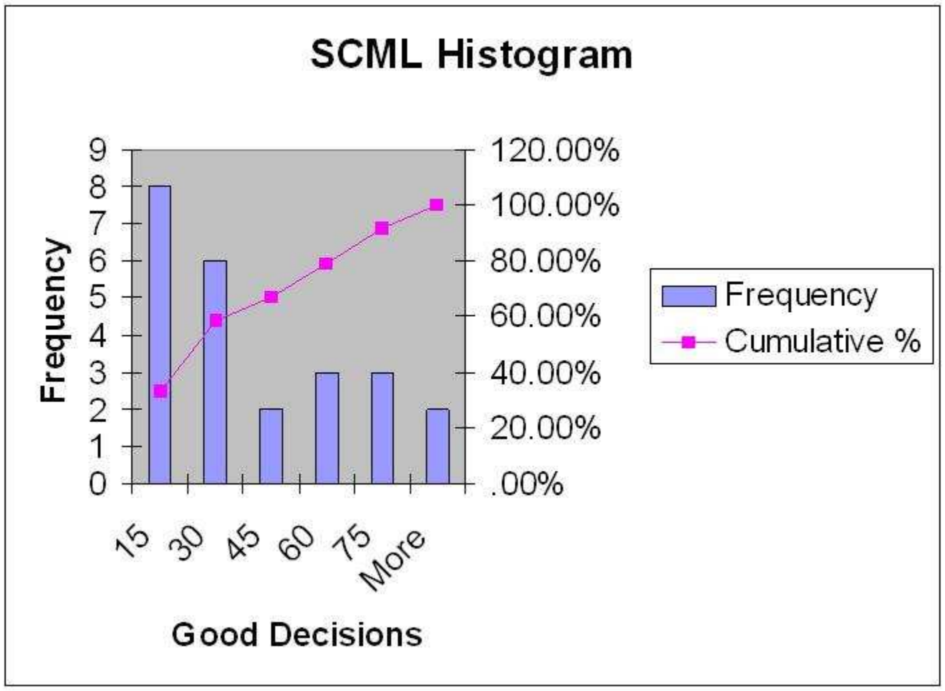
Table 29: SCML Decision Statistics

Measures of variability depict how widely data in a sample varies from the mean, otherwise known as the spread of the data. The deviation is an actual measure of this difference, and is shown in the third column of Table 29. The standard deviation is the average deviation of each value from the mean. The Logfile group had a standard deviation of 16.98 decisions across its data points as opposed to the much higher SCML group standard deviation of 29.65. The range is a simple calculation of the difference between the highest and lowest values in group of data. Range values are 55 for the Logfile group and 107 for SCML group. For each value in the table, the corresponding Z-score is shown in the third column of information. A *Z-score* is a value representing number of standard deviations a data point is from the mean. For example, a value with a Z-score of -1.22 is 1.22 standard deviations to the left of (less) than the mean.

Examining the relative frequencies of decision-making will give an even better depiction of how the participants in each group fared during the experiment. Figure 61 are histograms of frequency data for both experimental groups. Each bin in the frequency table is a count of the number of participants that fell with a certain range. For example, Bin 15 is a count of all players with between 0 and 15 good decisions; Bin 30 are all players with between 16 and 30 good decisions. Referring to the upper table in the diagram, which is for the Logfile group, it can be seen that there were more players who had fifteen or fewer decisions than any other category. This is also true for the SCML group. However, the cumulative affect is different, that is, the percentage of total players represented by the bins. The histograms show the distribution of decision counts as a cumulative effect. Eighty-eight percent of the participants on the Logfile group had 45 or fewer good decisions, while one-third of the SCML group had more that 45 good decisions.



(a)



(b)

Figure 61: Decision Frequency Histograms

6.4 Hypothesis Testing

Hypothesis testing is the process of using statistical analysis and probability distributions to determine the validity of a claim. This is done by analyzing two competing claims: a null hypothesis and an alternate hypothesis. A *null hypothesis* is a theory that is put forward as a basis for an argument. The alternative hypothesis is a statement of what is really to be proven. The results of the experiment are used to disprove or reject the null hypothesis, validating the alternative. The hypotheses are stated by comparing sampling statistics, such as the mean, standard deviation, or variance.

For this experiment, the hypothesis testing revolves around a comparison of the sample means. The mean (Greek symbol μ) is the average number of good decisions in a group. The mean for the Logfile group is depicted in the analysis with μ. The SCML group's mean is shown as μ_0. The null hypothesis, H_0, is that there is no difference between the means and the probabilities for either group's success are the same. H_0 means that the SCML-encoded information had no affect on the decision-making. Mathematically, H_0 is $\mu = \mu_0$. The alternate hypothesis, H_a is that there is a difference between the means of the two groups, given by the formula: $\mu \neq \mu_0$.

A normal probability distribution curve was created with the sample statistics in order to compare the hypotheses. We can assume a normal distribution for the data because of the central limit theorem. The *central limit theorem* states that the distribution of an average tends to be normal, even when the distribution from which the average is computed is non-normal [59]. A test statistic can be calculated from the sample data and plotted on a normal distribution to determine which hypothesis to accept. The technique for comparing the decision samples is called a z-test for sample means. The test statistic is represented as z. The process is to calculate the z value and determine where it occurs on the probability curve.

A level of significance, (denoted α) has to be chosen to establish the probability of error. There are two kinds of errors to be considered: *Type I* and *Type II.* A *Type I* error (also called *errors of the first kind*) occurs when the null hypothesis is rejected when it is true. *Type II* errors (*errors of the second kind*) denote when the alternative hypothesis is wrongly rejected. Type I errors are the most important to avoid, so the α value is selected as small as possible. For this experiment, $\alpha = 0.15$, meaning that there is a 15% probability of a Type I error. Using this value, a confidence interval under the distribution curve can be determined. The confidence interval refers to the range of values for the test statistics where the null hypothesis can be safely accepted. The critical region (also known as the region of rejection) are the set of values for which the null hypothesis should be rejected.

The following table (Table 30) shows the results of a Z-test on the means of both experiment groups. The mean and number of observations are taken directly from the sampling data of both experimental groups. P-values are calculated using standard statistical formulas. A *P-value* is the probability of getting a value of a test statistic as extreme or more extreme than that observed by chance alone, if the null hypothesis is true. In the table are p-values for one-tailed and two-tailed hypothesis test. A *one-tailed test* is a statistical test where the values for which we can reject the null hypothesis occurs on one-side of the distribution curve. *Two-tailed tests* have rejection regions on either side of the distribution curve. Because H_a states that there will be a difference in the means, but does not state which sample will have the higher μ value, the two-tailed test is the relevant p-value.

Table 30: z-Test Comparison of Sample Means

	Logfile	*SCML*
Mean	24.64	34.38
Observations	25	24
Hypothesized Mean Difference	0	
z	-1.402	
P(Z<=z) one-tail	0.0803	
z-critical one-tail	0.8416	
P(Z<=z) two-tail	0.1607	
z-critical two-tail	1.2816	

The z-critical two-tail, the critical value, is plus or minus 1.28 as shown in the table. A critical value is the threshold for which the value of the z-test should be rejected. The calculated z-statistic falls outside the region of acceptance to the left of the distribution curve, as depicted in Figure 62 [8]. This z value means that there is a 84% probability that the alternate hypothesis, H_a, is true and there is a statistically significant difference in number of good decisions between the two groups. The fact that z occurs on the left in the two-tailed test means that $\mu < \mu_0$; there is a statistically significant probability that the average number of good decisions will be higher for the SCML group than for the Logfile group.

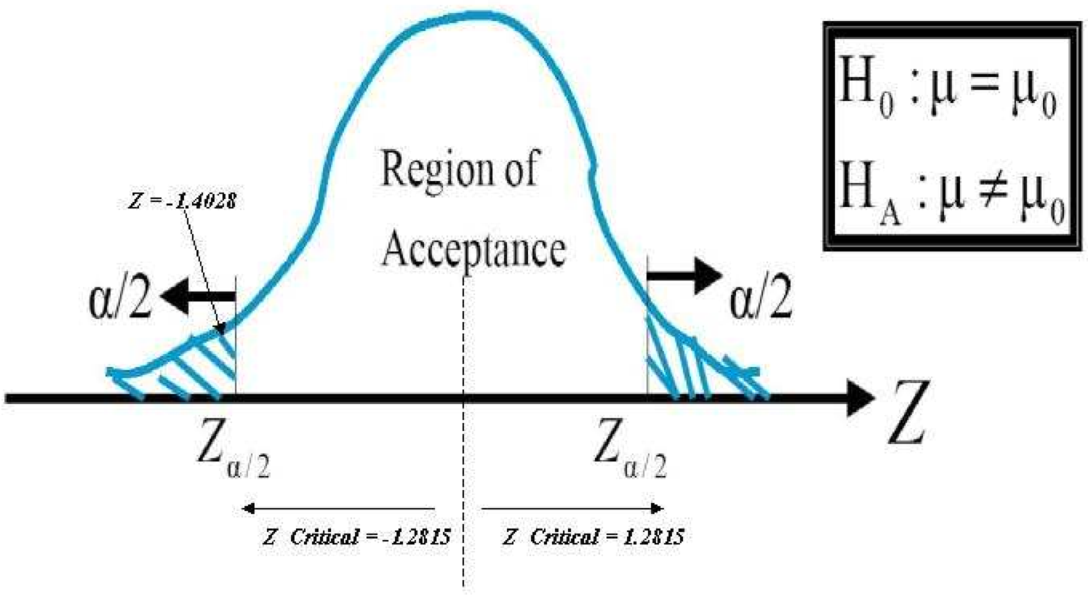


Figure 62: Z-statistic on distribution curve

I also performed a t-test for sample means to validate this result. A t-test is useful for approximating the normal distribution if the number of data values is less than thirty. There were enough data values that a z-test could be used, however, it helps to check the results using another statistical test. In Table 31, the calculated t-statistic is -1.5, which is outside the confidence interval established at the t-critical (two-tail) value of ±1.489. The calculate P-value means that there is an 85.5% probability that the SCML group will have the better average of good decisions between the two experimental groups.

Table 31: t-Test for Sample Means

	*L*ogfile	*S*CML
Mean	24.041	34.375
Observations	24	24
Pearson Correlation	0.040	
Hypothesized Mean Difference	0	
df	23	
t Stat	-1.506	
P(T<=t) one-tail	0.073	
t Critical one-tail	1.060	
P(T<=t) two-tail	0.145	
t Critical two-tail	1.489	

6.5 Summary

The results of the experiment were not affected by the demographics or level of experience of the participants in either group. The performance metrics support the conclusion that the SCML group made more good decisions about the maze during game play. There is a statistically significant improvement in the amount of good decision-making for the group that used SCML-encoded information for planning. On the average, there were 30% more of the good decisions from the SCML group.

Chapter 7

CONCLUSION

I feel that the greatest reward for doing is the opportunity to do more.

– Dr. Jonas Salk

7.1 Chapter Synopsis

Chapter 1 establishes the motivation and problem space for the scenario research. Chapter 2 is background information on narrative forms and their commonalities. Chapter 3 describes the research approach of defining a scenario framework based on conceptually modeling narrative and defining explicit relationships between scenario elements. Chapter 4 highlights potential uses for the scenario framework and presents a proof-of-concept scenario generator tool called Aesop. Chapter 5 is the design of an experiment to validate the positive impact of a computer readable scenario format on decision making. Chapter 6 presented the results of the empirical study comparing a sample of decision-makers using a simulation game.

7.2 Research Contribution

The primary intellectual contribution of the research is the scenario ontology and representation. The ontology was derived from an extensive examination of narrative domains, identifying the teleological story components, their relationships, constraints, and establishing the semantics for them. The ontology includes the scenario conceptual model, dynamic navigation model, and mathematical description of scenario elements and properties.

The secondary intellectual contribution is the hyperscenario framework which includes the scenario grammar and design and implementation of a scenario specification language based on the scenario ontology, SCML. The use of a markup language as the implementation platform effectively separates the form of a scenario from the scenario applications that manipulate it.

7.3 Future Work

There are three major tasks for the hyperscenario framework: 1) creating an XMLS version of SCML, 2) Mapping the scenario ontology to DAML+OIL. 3) Performing an empirical study on the impact of scenario navigation styles on decision-making.

7.3.1 Defining and implementing and XMLS version of SCML

As described in previous chapters, the SCML DTD was chosen for implementation based on existing tools and technologies available at the beginning of this effort. However, the XML Schema definition has now matured and there are now a wealth of tool sets that can effectively use XMLS. Some of the desirable features of XMLS include:

- Object-Oriented Approach
- Wider range of basic types available, more than the CDATA and PCDATA types in DTDs. The basic types include byte, integer, string and floating point numbers as well as ISO types for internationalization.
- The ability to create complex, user-defined types from the basic types and other user defined types
- Assignment of cardinality constraints and multiplicities to data
- The ability to require uniqueness of elements
- XMLS is written in XML
- It is easier to associate with both relational and object-oriented databases, as well as XML-oriented data storage structures. XQuery and XForms technologies for distributed information storage and retrieval assume an XML Schema

The new version of the scenario language, called SCML-S, will support strong data typing in the language and better support for automated reasoning through interaction with RDF and DAML. It will also be possible to have stand-alone scenario elements, such as episodes and events, since there is not the constraint of a root element.

7.3.2 Creating a DAML+OIL implementation of the scenario ontology

The DAML+OIL language, and its successor, OWL-S, are the basis for the semantic web. Software agents that exchange information across networks are being coded to lookup and access services based on DAML+OIL syntax and rules. SCML as an implementation suffers from the limitations of being a DTD, as outlined in the previous section. There is also a problem that it is difficult to represent the axioms and rules of a domain in an XML language. A DAML+OIL version of the scenario ontology would allow the inclusion of descriptive logics into the mathematical model, making scenario element relationships more explicit and easier to define. Automated reasoning support could then be created for the scenario framework, making it possible to build tools that not only display stories, but could "reason" about stories as well. The DAML+OIL representation of the scenario framework would also have have the aforementioned affect of making the ontology in a better form for mapping to a wide range of existing ontologies.

7.3.3 Empirical study of Scenario Navigation for Decision-making

The hyperlink structures and navigational styles outlined in the framework chapter were not part of the decision-making experiment. To keep the interaction simple, each scenario was contained in only one SCML document. The collaborative and distributed nature of hyperscenarios need to be tested in an environment that supports creating scenario artifacts that are linked together to create a coherent story. The experiment will have to be concerned with the type of collaborative activities that can be supported with a distributed scenario and how can the story be managed and maintain coherency. Can software agents be constructed that can follow the distributed links of a hyperscenario to create several versions of a story?

7.4 Conclusion

Scenarios are narratives that help policy makers and system designers choose among alternative courses of action. Scenarios also appear to mimic our cognitive processes, making a story view of information fit naturally into problem solving. Scenario-based decision-making crosses many domains and multiple perspectives. Existing scenario environments are not capable of supporting the wide spectrum of formality from executable simulation programs to free-form text to streaming media descriptions.

The thesis of this research claimed that a computer readable scenario framework could capture the semantics of a problem domain and make scenarios an active part of decision making. This dissertation described the Hyperscenario Framework and a scenario ontology derived by examining alternate forms of narrative. The approach was to define a scenario conceptual model based on the forms of narrative and the activities of storytelling. This method separates what a narrative is from how it is used. This research contributed an ontology that included a conceptual model of scenarios, a formal mathematical model, a context-free grammar, and a language implementation of that grammar. The results of a web-enabled simulation experiment validated the claim of improvement in the number of good decisions using SCML-encoded documents. Future work will involve improving the language implementation, representing the ontology in a computer readable form, and studying the impact of scenario navigation on decision-making.

Appendix A

SCENARIO MARKUP LANGUAGE DOCUMENT TYPE DEFINITION

```
<!-- SCML(Scenario Markup Language) -->
<!-- Revised: Mar 2005              -->
<!-- Author: Reginald L. Hobbs      -->

<!ENTITY % actiontp "atrans|ptrans|propel|move|grasp|ingest|
expel|mtrans|conc|mbuild|attend|speak">
<!ENTITY % acttp "setup|conflict|resolution|denouement">
<!ENTITY % cuttp "reaction|jump">

<!ELEMENT hyperscenario (goal, cast?, inventory?, (episode+|act+))>
<!ATTLIST hyperscenario
     hyperscenarioID ID #IMPLIED
     title NMTOKEN #REQUIRED
     logline CDATA #IMPLIED
     synopsis CDATA #IMPLIED
     settings CDATA #IMPLIED>

<!ELEMENT cast (character+)>

<!ELEMENT inventory (prop+)>

<!ELEMENT character (role|actor|prop)>
<!ELEMENT role (goal,(archetype?|actor+))>
<!ATTLIST role (goal+)
     roleID ID #REQUIRED
     name NMTOKEN #REQUIRED
     actions CDATA #IMPLIED>

<!ELEMENT actor (#PCDATA)>
```

```
<!ATTLIST actor

     actorID ID #REQUIRED

     name NMTOKEN #REQUIRED

     role CDATA #IMPLIED

     responsibility IDREF #IMPLIED>


<!ELEMENT prop (#PCDATA)>

<!ATTLIST prop

     propID ID #REQUIRED

     name NMTOKEN #REQUIRED

     targetOF IDREF #IMPLIED>


<!ELEMENT act (episode+)>

<!ATTLIST act

     name NMTOKEN #IMPLIED

     acttp (%acttp;) "conflict">




<!ELEMENT episode (goal, (scene+|event+))>

<!ATTLIST episode

     episodeID ID #IMPLIED

     name NMTOKEN #IMPLIED>


<!ELEMENT goal (#PCDATA)>

<!ATTLIST goal

     goalID ID #IMPLIED

     name NMTOKEN #IMPLIED

     description CDATA #IMPLIED>
```

```
<!ELEMENT scene (setting, event+)>
<ATTLIST scene
    sceneID ID #IMPLIED
    name NMTOKEN #IMPLIED
    slugline CDATA #IMPLIED>

<!ELEMENT setting (#PCDATA)>

<!ELEMENT event (goal?, action+)>
<!ATTLIST event
     eventID ID #IMPLIED
     name NMTOKEN #IMPLIED>

<!ELEMENT action (goal?, actor?, prop?, storylink*)>
<!ATTLIST action
     actionID ID #IMPLIED
     name NMTOKEN #IMPLIED
     actiontype (%actiontp;) "move"
     originator IDREF #IMPLIED
     target IDREF #IMPLIED>

<!ELEMENT storylink (annotation|flashback|transition|
     cut|slide|dependency|equivalence|prerequisite|
     rewind|setup|conclusion)>
<!ATTLIST storylink
     source ID #IMPLIED
     target IDREF #REQUIRED
     class CDATA #IMPLIED
     xml:link CDATA #FIXED "simple"
     href CDATA #IMPLIED
     role CDATA "transition"
     title CDATA #IMPLIED
     show (embed|replace|new) "new"
     actuate (auto|user) "user"
     behavior CDATA #IMPLIED>

<!ELEMENT annotation EMPTY>
<!ELEMENT flashback EMPTY>
<!ELEMENT transition EMPTY>
<!ELEMENT cut EMPTY>
<!ATTLIST cut
     cuttype (%cuttp;) "reaction">

<!ELEMENT slide EMPTY>
<!ELEMENT dependency EMPTY>
<!ELEMENT equivalence EMPTY>
<!ELEMENT prerequisite EMPTY>
<!ELEMENT rewind EMPTY>
<!ELEMENT setup EMPTY>
<!ELEMENT conclusion EMPTY>
```

Appendix B

EXPERIMENT DOCUMENTS

Call for Research Participants

I am seeking participants to use an experimental game environment developed as part of my research. The game is a 3-D maze simulation accessible across the web. The purpose of the game is to determine how the availability of a narrative-based, scenario view of data impacts decision-making.

Using the arrow keys and function keys on your keyboard, you will navigate and interact with the maze environment. Each participant will play the game several times, using log file information generated between sessions for planning. The experiment should take 70 minutes or less to complete. Because of incompatibilities with the MS Internet Explorer Java environment, the experiment has to be accessed using Mozilla 4.0 or Netscape 7.0 or higher.

This is a research study, so you will be required to complete a short, online survey at the end of the game play (all information is confidential and protected by IRB guidelines). If you are interested you may access further experiment information at:
http://www.cc.gatech.edu/~reggie/experiment.htm

Thank You.
Reginald L. Hobbs
reggie@cc.gatech.edu

Research Consent Form

Georgia Institute of Technology

1. **Project Title**: The Effect of a Computational Scenario Framework on Decision-making in a 3-D Maze Simulation

2. **Investigator:** Reginald L. Hobbs

3. **Purpose of Research:** You are being asked to be a volunteer in a research study to assess the impact of a scenario specification environment on decision-making. The experiment will take approximately 70 minutes to complete. There will be a maximum of 50 participants in this study.

4. **Procedures:** The experiment will consist of the repetition of a series of game playing sessions within a web-enabled 3-D maze. There will be six game sessions per participant. The first game is done for initialization purposes and for game familiarization; at that time you will be presented with an instruction page that describes how to play the game. The length of individual sessions will be limited to 5 minutes or until the objective is achieved, whichever occurs first. You will be randomly assigned to one of two groups that will have alternate methods of reviewing game sessions. A browser will be available to review the previous game for the purpose of planning the next session. Intra-game reviews will be limited to 5 minutes or less. A survey will be presented to you at the end of the overall session to collect your assessment of the game.

5. **Foreseeable Risks/Discomforts:** The risks involved are minimal and are no greater than those involved in accessing the Internet, browsing the Web, or playing simple puzzle games.

6. **Benefits:** You may not benefit directly from participating in this study; however, your participation may provide additional knowledge about the domain being investigated.

7. **Compensation:** You will not be paid for your participation in this research study.

8. **Confidentiality:** The following procedures will be followed to keep your personal information confidential in this study: The data that is collected about you will be kept private to the extent allowed by law. To protect your privacy, your records will be kept under a code number rather than by name. Your name and any other fact that might point to you will not appear when results of this study are presented or published.

   You should be aware that the experiment is not being run from a secure Web server of the kind typically used to handle credit card transactions, so there is a small possibility that responses could be viewed by unauthorized third parties (e.g., computer hackers). Also, in general the web page software will log as header lines the IP address of the machine you use to access this page, but otherwise no other information will be stored unless you explicitly enter it.

   To make sure that this research is being carried out in the proper way, the Georgia Institute of Technology IRB may review study records.

9. **Costs:** There is no cost to you for participation in this study.

10. **Subject Rights:** Your participation in this study is voluntary. You do not have to be in this study if you don't want to be. You have the right to change your mind and leave the study at any time without giving any reason, and without penalty. Any new information that may make you change your mind about being in this study will be given to you. You do not waive any of your legal rights by acknowledging this consent form.

Questions about the Study or Your Rights as a Research Subject: If you have any questions about the study, you may contact Reginald Hobbs, at telephone (404) 894-1807. If you have any questions about your rights as a research subject, you may contact Ms. Alice Basler, Georgia Institute of Technology at (404) 894-6942.

Registration Form

1. Name

2. E-mail address. Please enter a complete e-mail address.

3. Age

   - 18-25
   - 25-35
   - 35-45
   - 45+

4. Gender

   - Male
   - Female

5. Educational Level

   - Undergraduate
   - Graduate
   - Post-graduate

6. Please indicate your level of experience with the following games/environments on a scale of 1 to 5. 1=never, 3=sometimes, 5=often.

Role-Playing Games	1	2	3	4	5
Crossword Puzzles	1	2	3	4	5
Word Search	1	2	3	4	5
Logic Puzzles	1	2	3	4	5
Chat Rooms	1	2	3	4	5
PC Simulation Games	1	2	3	4	5
Adventure/Combat MUDs	1	2	3	4	5
Social MUDs	1	2	3	4	5
Video Games	1	2	3	4	5
Board Games	1	2	3	4	5
Card Games	1	2	3	4	5

Comments on Registration Form

1. **Name**: For identification purposes. The first initial and last name will be used to create a username for access to the simulation environment Participants may access the system only once.

2. **E-mail Address**: Verification of identity to prevent duplication.

3. **Age**: Demographic purposes.

4. **Gender**: Demographic purposes.

5. **Educational Level**: Demographic purposes.

6. **Game Experience**: The categories were chosen to determine the type and frequency of experience with the following attributes:

    - Immersive/Role-Playing
    - Games of Chance
    - Pattern Matching
    - Memory
    - Interaction

Appendix C

SCENARIO GRAMMAR AUTOMATON

```
State 27 conflicts: 2 shift/reduce

Grammar

    0 $accept: scenario $end

    1 scenario: goal cast inventory episodes

    2         | goal cast inventory acts

    3 cast: /* empty */

    4     | cast character

    5 character: prop

    6          | role

    7          | actor

    8 inventory: /* empty */

    9          | manifest props

   10 props: prop

   11      | props prop

   12 role: archetype

   13 acts: act

   14     | acts act

   15 act: episodes intermission

   16 episodes: episode

   17         | episodes episode

   18 episode: goal events

   19        | goal scenes

   20 scenes: scene

   21       | scenes scene

   22 scene: setting events

   23 events: event

   24       | events event

   25 event: actions
   26 actions: action

   27        | actions action

   28 action: verb

   29       | verb prop

   30       | actor verb

   31       | actor verb prop



Terminals, with rules where they appear

$end (0) 0

error (256)

goal (258) 1 2 18 19

prop (259) 5 10 11 29 31

actor (260) 7 30 31

archetype (261) 12

setting (262) 22

intermission (263) 15

manifest (264) 9

verb (265) 28 29 30 31


Nonterminals, with rules where they appear

$accept (11)

    on left: 0

scenario (12)

    on left: 1 2, on right: 0

cast (13)

    on left: 3 4, on right: 1 2 4

character (14)

    on left: 5 6 7, on right: 4
```

```
inventory (15)

    on left: 8 9, on right: 1 2

props (16)

    on left: 10 11, on right: 9 11

role (17)

    on left: 12, on right: 6

acts (18)

    on left: 13 14, on right: 2 14

act (19)

    on left: 15, on right: 13 14

episodes (20)

    on left: 16 17, on right: 1 15 17

episode (21)

    on left: 18 19, on right: 16 17

scenes (22)

    on left: 20 21, on right: 19 21

scene (23)

    on left: 22, on right: 20 21

events (24)

    on left: 23 24, on right: 18 22 24

event (25)

    on left: 25, on right: 23 24

actions (26)

    on left: 26 27, on right: 25 27

action (27)

    on left: 28 29 30 31, on right: 26 27
```

```
States


state 0

    0 $accept: . scenario $end

    1 scenario: . goal cast inventory episodes

    2         | . goal cast inventory acts

    goal  shift, and go to state 1

    scenario  go to state 2

state 1

    1 scenario: goal . cast inventory episodes

    2         | goal . cast inventory acts

    3 cast: .

    4     | . cast character

    $default  reduce using rule 3 (cast)

    cast  go to state 3

state 2

    0 $accept: scenario . $end

    $end  shift, and go to state 4

state 3

    1 scenario: goal cast . inventory episodes

    2         | goal cast . inventory acts

    4 cast: cast . character

    5 character: . prop

    6          | . role

    7          | . actor

    8 inventory: .  [goal]

    9          | . manifest props

   12 role: . archetype

    prop       shift, and go to state 5
    actor      shift, and go to state 6

    archetype  shift, and go to state 7

    manifest   shift, and go to state 8

    $default  reduce using rule 8 (inventory)

    character  go to state 9

    inventory  go to state 10

    role       go to state 11

state 4

    0 $accept: scenario $end .

    $default  accept

state 5

    5 character: prop .

    $default  reduce using rule 5 (character)

state 6

    7 character: actor .

    $default  reduce using rule 7 (character)

state 7

   12 role: archetype .

    $default  reduce using rule 12 (role)

state 8

    9 inventory: manifest . props

   10 props: . prop

   11      | . props prop

    prop  shift, and go to state 12

    props  go to state 13

state 9

    4 cast: cast character .

    $default  reduce using rule 4 (cast)

state 10
```

```
    1 scenario: goal cast inventory . episodes
    2         | goal cast inventory . acts
   13 acts: . act
   14     | . acts act
   15 act: . episodes intermission
   16 episodes: . episode
   17         | . episodes episode
   18 episode: . goal events
   19        | . goal scenes
    goal  shift, and go to state 14
    acts      go to state 15
    act       go to state 16
    episodes  go to state 17
    episode   go to state 18
state 11
    6 character: role .
    $default  reduce using rule 6 (character)
state 12
   10 props: prop .
    $default  reduce using rule 10 (props)
state 13
    9 inventory: manifest props .  [goal]
   11 props: props . prop
    prop  shift, and go to state 19
    $default  reduce using rule 9 (inventory)
state 14
   18 episode: goal . events
   19        | goal . scenes
   20 scenes: . scene
   21       | . scenes scene

   22 scene: . setting events

   23 events: . event

   24       | . events event

   25 event: . actions

   26 actions: . action

   27        | . actions action

   28 action: . verb

   29       | . verb prop

   30       | . actor verb

   31       | . actor verb prop

    actor    shift, and go to state 20

    setting  shift, and go to state 21

    verb     shift, and go to state 22

    scenes   go to state 23

    scene    go to state 24

    events   go to state 25

    event    go to state 26

    actions  go to state 27

    action   go to state 28

state 15

    2 scenario: goal cast inventory acts .  [$end]

   14 acts: acts . act

   15 act: . episodes intermission

   16 episodes: . episode

   17         | . episodes episode

   18 episode: . goal events

   19        | . goal scenes

    goal  shift, and go to state 14
    $default  reduce using rule 2 (scenario)

    act       go to state 29

    episodes  go to state 30

    episode   go to state 18

state 16

   13 acts: act .

    $default  reduce using rule 13 (acts)

state 17

    1 scenario: goal cast inventory episodes .  [$end]

   15 act: episodes . intermission

   17 episodes: episodes . episode

   18 episode: . goal events

   19        | . goal scenes

    goal          shift, and go to state 14

    intermission  shift, and go to state 31

    $default  reduce using rule 1 (scenario)

    episode  go to state 32

state 18

   16 episodes: episode .

    $default  reduce using rule 16 (episodes)

state 19

   11 props: props prop .

    $default  reduce using rule 11 (props)

state 20

   30 action: actor . verb

   31       | actor . verb prop

    verb  shift, and go to state 33

state 21

   22 scene: setting . events
   23 events: . event

   24       | . events event

   25 event: . actions

   26 actions: . action

   27        | . actions action

   28 action: . verb

   29       | . verb prop

   30       | . actor verb

   31       | . actor verb prop

    actor  shift, and go to state 20

    verb   shift, and go to state 22

    events   go to state 34

    event    go to state 26

    actions  go to state 27

    action   go to state 28

state 22

   28 action: verb .  [$end, goal, actor, setting, intermission, verb]

   29       | verb . prop

    prop  shift, and go to state 35

    $default  reduce using rule 28 (action)

state 23

   19 episode: goal scenes .  [$end, goal, intermission]

   21 scenes: scenes . scene

   22 scene: . setting events

    setting  shift, and go to state 21

    $default  reduce using rule 19 (episode)

    scene  go to state 36

state 24

   20 scenes: scene .
```

```
    $default  reduce using rule 20 (scenes)

state 25

   18 episode: goal events .  [$end, goal, intermission]

   24 events: events . event

   25 event: . actions

   26 actions: . action

   27        | . actions action

   28 action: . verb

   29       | . verb prop

   30       | . actor verb

   31       | . actor verb prop

    actor  shift, and go to state 20

    verb   shift, and go to state 22

    $default  reduce using rule 18 (episode)

    event    go to state 37

    actions  go to state 27

    action   go to state 28

state 26

   23 events: event .

    $default  reduce using rule 23 (events)

state 27

   25 event: actions .  [$end, goal, actor, setting, intermission, verb]

   27 actions: actions . action

   28 action: . verb

   29       | . verb prop

   30       | . actor verb

   31       | . actor verb prop

    actor  shift, and go to state 20

    verb   shift, and go to state 22
    actor     [reduce using rule 25 (event)]

    verb      [reduce using rule 25 (event)]

    $default  reduce using rule 25 (event)

    action  go to state 38

state 28

   26 actions: action .

    $default  reduce using rule 26 (actions)

state 29

   14 acts: acts act .

    $default  reduce using rule 14 (acts)

state 30

   15 act: episodes . intermission

   17 episodes: episodes . episode

   18 episode: . goal events

   19        | . goal scenes

    goal          shift, and go to state 14

    intermission  shift, and go to state 31

    episode  go to state 32

state 31

   15 act: episodes intermission .

    $default  reduce using rule 15 (act)

state 32

   17 episodes: episodes episode .

    $default  reduce using rule 17 (episodes)

state 33

   30 action: actor verb .  [$end, goal, actor, setting, intermission, verb]

   31       | actor verb . prop

    prop  shift, and go to state 39

    $default  reduce using rule 30 (action)
```

```
state 34

   22 scene: setting events .  [$end, goal, setting, intermission]

   24 events: events . event

   25 event: . actions

   26 actions: . action

   27        | . actions action

   28 action: . verb

   29       | . verb prop

   30       | . actor verb

   31       | . actor verb prop

    actor  shift, and go to state 20

    verb   shift, and go to state 22

    $default  reduce using rule 22 (scene)

    event    go to state 37

    actions  go to state 27

    action   go to state 28

state 35

   29 action: verb prop .

    $default  reduce using rule 29 (action)

state 36

   21 scenes: scenes scene .

    $default  reduce using rule 21 (scenes)

state 37

   24 events: events event .

    $default  reduce using rule 24 (events)

state 38

   27 actions: actions action .

    $default  reduce using rule 27 (actions)

state 39
```

```
31 action: actor verb prop .

 $default  reduce using rule 31 (action)
```

Appendix D

DESIGN DOCUMENTATION

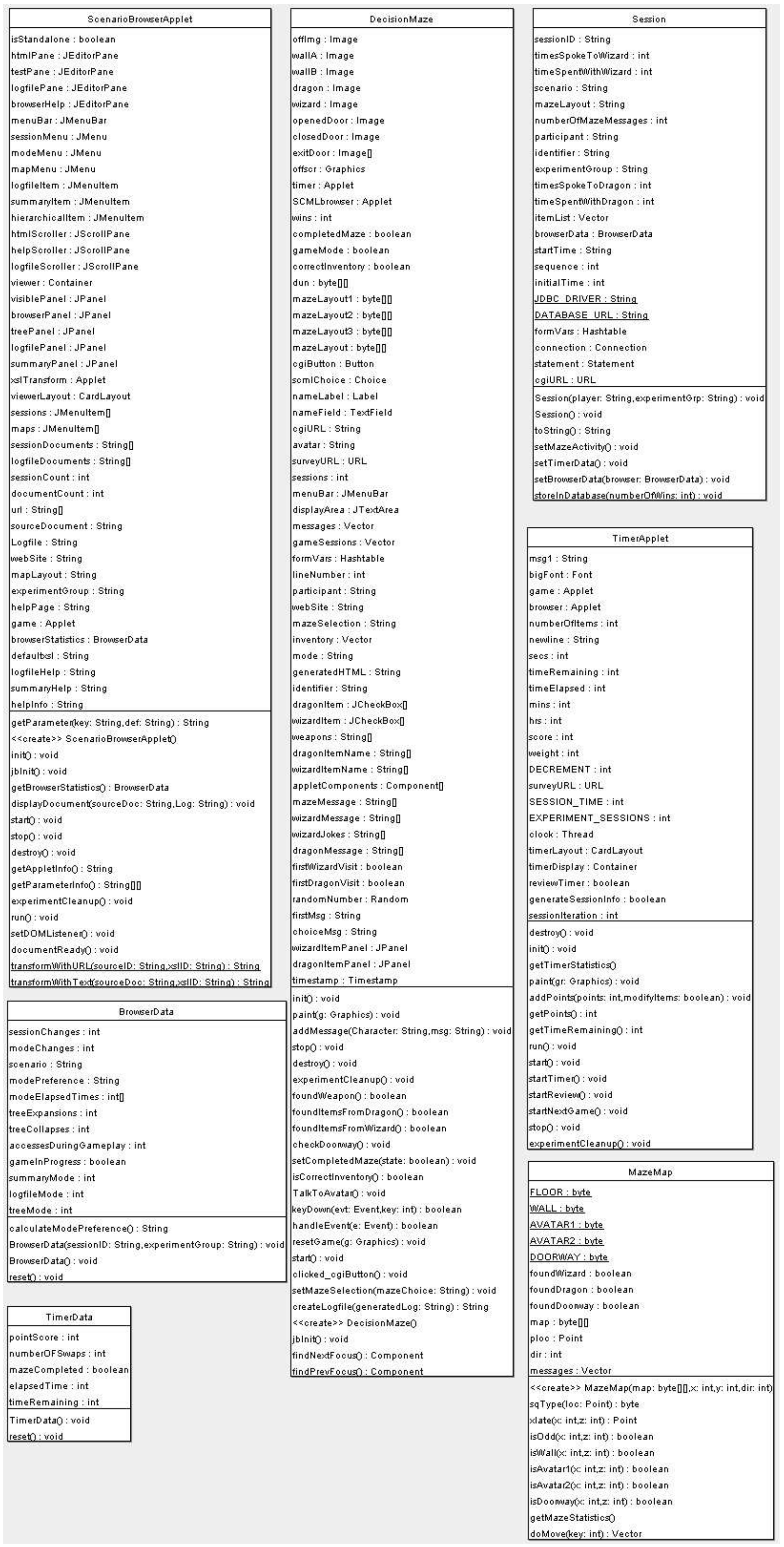


Figure 63: Decision Maze Experiment Class Diagrams

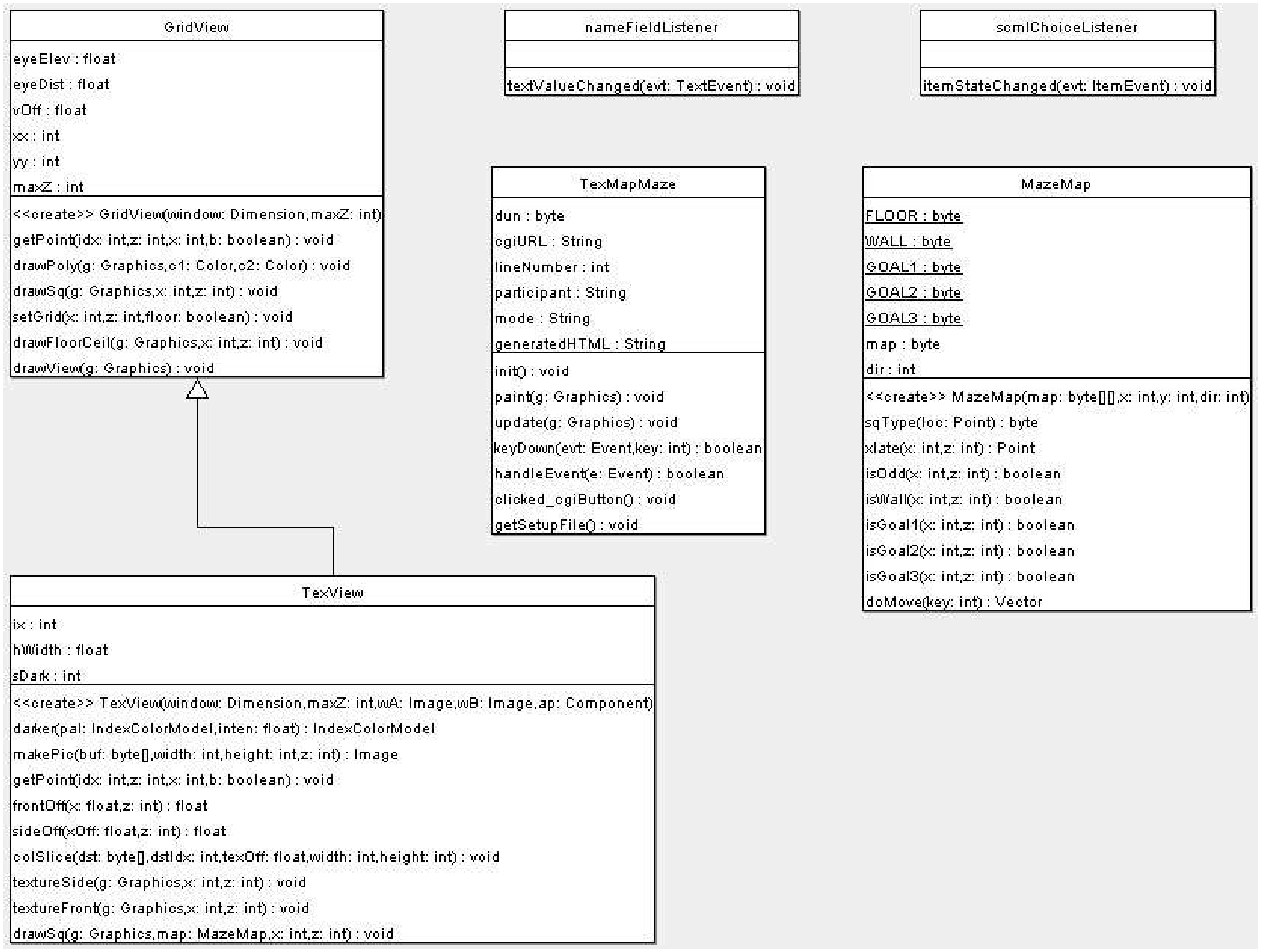

GridView
eyeElev : float
eyeDist : float
vOff : float
xx : int
yy : int
maxZ : int
<<create>> GridView(window: Dimension,maxZ: int)
getPoint(idx: int,z: int,x: int,b: boolean) : void
drawPoly(g: Graphics,c1: Color,c2: Color) : void
drawSq(g: Graphics,x: int,z: int) : void
setGrid(x: int,z: int,floor: boolean) : void
drawFloorCeil(g: Graphics,x: int,z: int) : void
drawView(g: Graphics) : void
nameFieldListener
textValueChanged(evt: TextEvent) : void
scmlChoiceListener
itemStateChanged(evt: ItemEvent) : void
TexMapMaze
dun : byte
cgiURL : String
lineNumber : int
participant : String
mode : String
generatedHTML : String
init() : void
paint(g: Graphics) : void
update(g: Graphics) : void
keyDown(evt: Event,key: int) : boolean
handleEvent(e: Event) : boolean
clicked_cgiButton() : void
getSetupFile() : void
MazeMap
FLOOR : byte
WALL : byte
GOAL1 : byte
GOAL2 : byte
GOAL3 : byte
map : byte
dir : int
<<create>> MazeMap(map: byte[][],x: int,y: int,dir: int)
sqType(loc: Point) : byte
xlate(x: int,z: int) : Point
isOdd(x: int,z: int) : boolean
isWall(x: int,z: int) : boolean
isGoal1(x: int,z: int) : boolean
isGoal2(x: int,z: int) : boolean
isGoal3(x: int,z: int) : boolean
doMove(key: int) : Vector
TexView
ix : int
hWidth : float
sDark : int
<<create>> TexView(window: Dimension,maxZ: int,wA: Image,wB: Image,ap: Component)
darken(pal: IndexColorModel,inten: float) : IndexColorModel
makePic(buf: byte[],width: int,height: int,z: int) : Image
getPoint(idx: int,z: int,x: int,b: boolean) : void
frontOff(x: float,z: int) : float
sideOff(xOff: float,z: int) : float
colSlice(dst: byte[],dstIdx: int,texOff: float,width: int,height: int) : void
textureSide(g: Graphics,x: int,z: int) : void
textureFront(g: Graphics,x: int,z: int) : void
drawSq(g: Graphics,map: MazeMap,x: int,z: int) : void

Figure 64: 3-D Maze Class Diagrams

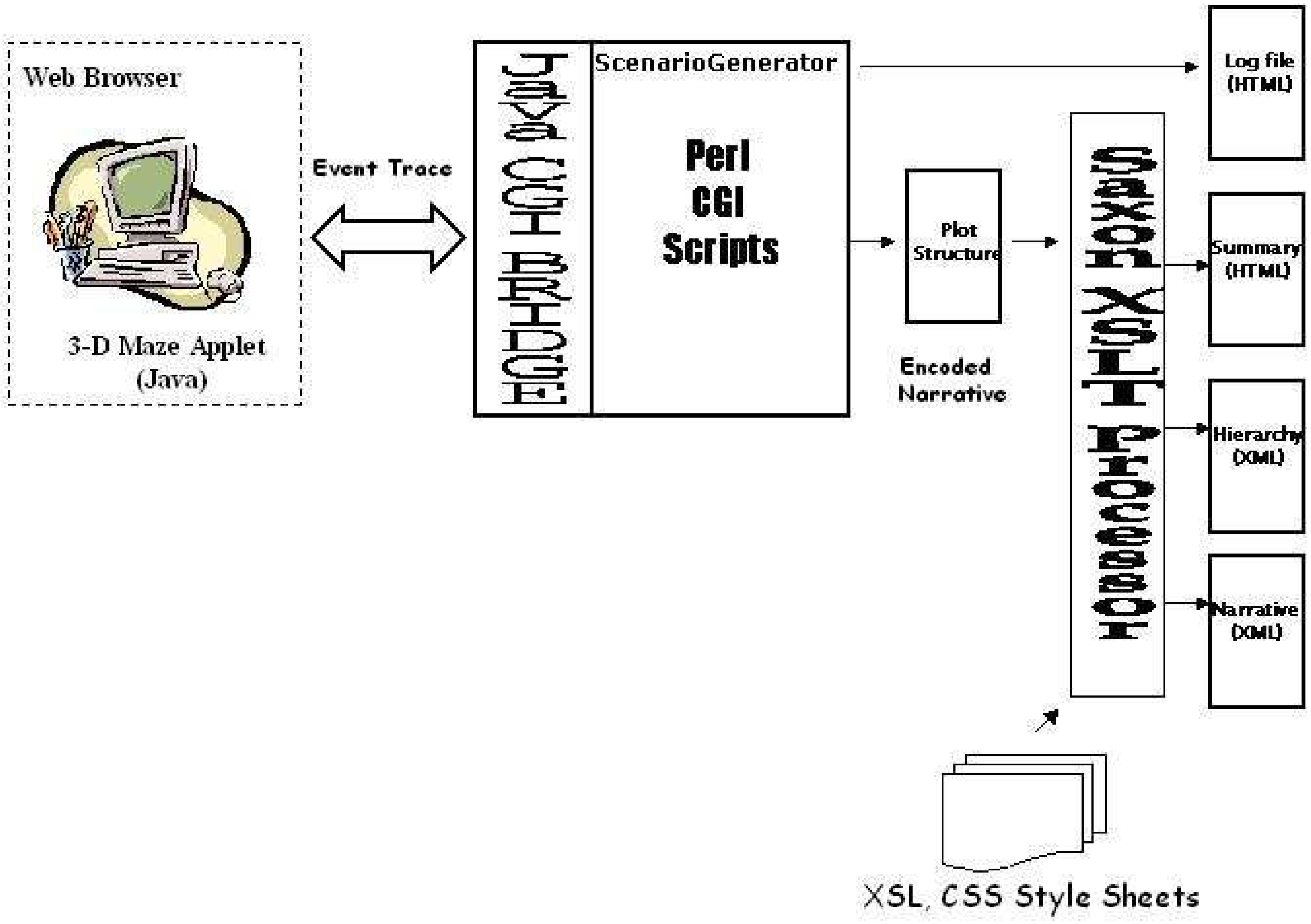


Figure 65: Scenario Prototype System Architecture

Appendix E

SHUTTLE SCENARIO

```
<hyperscenario title="Successful Shuttle Transport">
<goal>Process Passenger Request for Shuttle Transport</goal>
  <episode episodeID=1 name="Initiate Transport Request">
    <goal>Handle Transport Request</goal>
    <event eventID=1 name="Notify Shuttle Agent of Request">
      <action actionID=1 name="action1">
         <actor>BrokerAgent</actor>
      Send Order A Available Msg
         <storylink source="."><setup/></storylink>
         <storylink target="#action2"><transition/></storylink>
      </action>
    </event>
  </episode>
  <episode episodeID=2 name="Accept Bids For Transport Request">
    <goal>Accept Bid From Shuttle Agent</goal>
    <event eventID=2 name="Shuttle Agent Places Bid">
      <action actionID=2 name="action2">
        <actor>ShuttleAgent</actor>
      Calculate Offer
        <storylink target="#action1"><prerequisite/></storylink>
      </action>
      <action actionID=3 name="action3">
        <actor>ShuttleAgent</actor>
      Make Offer on Order A
        <storylink target="#action4"><transition/></storylink>
      </action>
    </event>
  </episode>
  <episode episode=3 name="Start Shuttle Simulation">
    <goal>Initialize Simulated Shuttle</goal>
    <event eventID=3 name="Activate Shuttle Simulation">
      <action actionID=4 name="action4">
        <actor>ShuttleAgent</actor>
      Send WakeUp Request
        <storylink target="#action3"><prerequisite/></storylink>
        <storylink target="#action5"><transition/></storylink>
      </action>
    </event>
  </episode>
  <episode episodeID=4 name="Process Multiple Transport Requests">
     <goal>Process Multiple Transport Requests</goal>
     <event  eventID=4 name="Notify Shuttle Agent of Request">
       <action actionID=5 name="action5">
         <actor>BrokerAgent</actor>
       Send Order B Available Msg
         <storylink target="#action4"><prerequisite/></storylink>
       </action>
     </event>
     <event  eventID=5 name="Shuttle Agent Places Bid">
       <action actionID=6 name="action6">
         <actor>ShuttleAgent</actor>
       Calculate Offer
        <storylink target="#action2"><flashback/></storylink>
       </action>
       <action actionID=7 name="action7">
         <actor>ShuttleAgent</actor>
       Send WakeUp Request
        <storylink target="#action8"><transition/></storylink>
      </action>
    </event>
  </episode>
  <episode episodeID=5 name="Choose Shuttle for Transport">
     <goal>Select Shuttle to Handle Request</goal>
     <event eventID=6 name="Shuttle Selection">
       <action actionID=8 name="action8">
         <actor>BrokerAgent</actor>
       Calculate Offers for Order A
         <storylink target="#action7"><prerequisite/></storylink>
       </action>
       <action actionID=9 name="action9">
         <actor>BrokerAgent</actor>
       Assign Order to Agent
       </action>
       <action actionID=10 name="action10">
         <actor>BrokerAgent</actor>
       Calculate Offers for Order B
         <storylink target="#action11"><transition/></storylink>
       </action>
     </event>
  </episode>
  <episode episodeID=6 name="Transport Passenger(s)">
     <goal>Transport Passenger(s) to Requested Destination</episode>
     <event eventID=7 name="Determine Transport Route">
       <action actionID=11 name="action11">
         <actor>ShuttleAgent</actor>
       Calculate Path
         <storylink target="#action10"><prerequisite/></storylink>
       </action>
       <action actionID=12 name="action12">
         <actor>ShuttleAgent</actor>
       Send Move Shuttle Request
       </action>
     </event>
     <event eventID=8 name="Shuttle Transport Movement">
       <action actionID=13 name="action13">
         <actor>Simulator</actor>
       Check Command
         <storylink target="."><annotation/></storylink>
       </action>
       <action actionID=14 name="action14">
         <actor>Simulator</actor>
       Send Shuttle Moving Msg
         <storylink target="."><annotation/></storylink>
       </action>
       <action actionID=15 name="action15">
         <actor>Simulator</actor>
       Send Shuttle Arrived Msg
         <storylink target="."><annotation/></storylink>
       </action>
     </event>
     <event eventID=9 name="Load Passenger(s)">
       <action actionID=16 name="action16">
         <actor>ShuttleAgent</actor>
       Send Load Shuttle for Order A Request
         <storylink target="#action18"><cut/></storylink>
       </action>
```

```
      <action actionID=17 name="action17">
        <actor>Simulator</actor>
      Check Command
        <storylink target="."><annotation/></storylink>
      </action>
      <action actionID=18 name="action18">
        <actor>ShuttleAgent</actor>
      Send Shuttle Loaded Msg
        <storylink target="#action19"><cut/></storylink>
      </action>
    </event>
    <event eventID=10 name="Unload Passenger(s)">
      <action actionID=19 name="action19">
        <actor>ShuttleAgent</actor>
      Send Unload Shuttle for Order A Request
      </action>
      <action actionID=20 name="action20">
        <actor>Simulator</actor>
      Check Command
        <storylink target="."><annotation/></storylink>
      </action>
      <action actionID=21 name="action21">
        <actor>Simulator</actor>
      Send Shuttle Unloaded Msg
        <storylink target="#action22"><transition/></storylink>
      </action>
    </event>
 </episode>
 <episode episodeID=7 name="Complete Transport Request">
```

```
    <goal>Assign Payment for Completed Transport</goal>

    <event eventID=11 name="Request Payment for Shuttle Transport">

      <action actionID=22 name="action22">

         <actor>ShuttleAgent</actor>

       Send Order A Invoice

         <storylink target="#action21"><prerequisite/></storylink>

         <storylink target="."><conclusion/></storylink>

       </action>

    </event>

  </episode>

</hyperscenario>
```

Appendix F

EMPIRICAL DATA

Table 32: Logfile Chat Room Experience Statistics

Mean	1.788
Standard Error	0.249
Median	1
Mode	1
Standard Deviation	1.431
Sample Variance	2.047
Kurtosis	1.561
Skewness	1.760
Range	4
Minimum	1
Maximum	5
Sum	59
Count	33
Confidence Level(95.0%)	0.507

Table 33: SCML Chat Room Experience Statistics

Mean	2.485
Standard Error	0.258
Median	2
Mode	1
Standard Deviation	1.482
Sample Variance	2.195
Kurtosis	-0.917
Skewness	0.676
Range	4
Minimum	1
Maximum	5
Sum	82
Count	33
Confidence Level(95.0%)	0.525

Table 34: Logfile Logic Puzzles Experience Statistics

Mean	3
Standard Error	0.200
Median	3
Mode	3
Standard Deviation	0.978
Sample Variance	0.957
Kurtosis	0.144
Skewness	-0.913
Range	3
Minimum	1
Maximum	4
Sum	72
Count	24
Confidence Level(95.0%)	0.413

Table 35: SCML Logic Puzzles Experience Statistics

Mean	2.606
Standard Error	0.162
Median	3
Mode	3
Standard Deviation	0.933
Sample Variance	0.871
Kurtosis	-0.642
Skewness	-0.329
Range	3
Minimum	1
Maximum	4
Sum	86
Count	33
Confidence Level(95.0%)	0.331

Table 36: Logfile Video Game Experience Statistics

Mean	3.042
Standard Error	0.310
Median	3
Mode	4
Standard Deviation	1.517
Sample Variance	2.303
Kurtosis	-1.450
Skewness	-0.158
Range	4
Minimum	1
Maximum	5
Sum	73
Count	24
Confidence Level(95.0%)	0.641

Table 37: SCML Video Game Experience Statistics

Mean	3.485
Standard Error	0.243
Median	4
Mode	5
Standard Deviation	1.395
Sample Variance	1.945
Kurtosis	-1.007
Skewness	-0.519
Range	4
Minimum	1
Maximum	5
Sum	115
Count	33
Confidence Level(95.0%)	0.495

Table 38: Logfile Board Game Experience Statistics

Mean	3.375
Standard Error	0.281
Median	3
Mode	5
Standard Deviation	1.377
Sample Variance	1.897
Kurtosis	-0.991
Skewness	-0.315
Range	4
Minimum	1
Maximum	5
Sum	81
Count	24
Confidence Level(95.0%)	0.582

Table 39: SCML Board Game Experience Statistics

Mean	3.424
Standard Error	0.200
Median	3
Mode	3
Standard Deviation	1.146
Sample Variance	1.314
Kurtosis	-0.872
Skewness	0.131
Range	4
Minimum	1
Maximum	5
Sum	113
Count	33
Confidence Level(95.0%)	0.407

Cumulative Frequency Tables

Logfile Decision

Frequency Distribution

Bin	Frequency	Cumulative %
15	11	44%
30	2	52%
45	9	88%
60	3	100%
75	0	100%
More	0	100%

SCML Decision

Frequency Distribution

Bin	Frequency	Cumulative %
15	8	33.3%
30	6	58.3%
45	2	66.7%
60	3	79.2%
75	3	91.7%
More	2	100.0%

Logfile Movement

Mean	**S**tdDev	**M**edian	**R**ange
119.00	94.84	111.5	389

Session ID	**M**oves	**D**eviation	**Z**-Score
1588430	177	58.00	0.49
9203300	105	-14.00	-0.12
3044340	265	146.00	1.23
1260580	19	-100.00	-0.84
3904260	49	-70.00	-0.59
3268210	81	-38.00	-0.32
4724100	4	-115.00	-0.97
1100230	391	272.00	2.29
3432480	2	-117.00	-0.98
2144250	155	36.00	0.30
2616130	33	-86.00	-0.72
1592150	115	-4.00	-0.03
2096480	57	-62.00	-0.52
3524360	87	-32.00	-0.27
3832220	166	47.00	0.39
3828330	253	134.00	1.13
3268500	183	64.00	0.54
3928900	5	-114.00	-0.96
2952200	49	-70.00	-0.59
1448130	108	-11.00	-0.09
1708180	179	60.00	0.50
2324500	130	11.00	0.09
3304540	147	28.00	0.24
2329000	137	18.00	0.15
3784900	173	54.00	0.45
3204120	261	142.00	1.19
1148330	3	-116.00	-0.97
3344410	49	-70.00	-0.59
3180160	181	62.00	0.52
2828370	6	-113.00	-0.95

SCML Movement

Mean	**S**tdDev	**M**edian	**R**ange
121.03	114.81	117.92	453

Session ID	**M**oves	**D**eviation	**Z**-Score
2272570	143	21.98	0.18
5361700	301	179.98	1.49
2984520	408	286.98	2.37
2988580	324	202.98	1.68
2500430	18	-103.03	-0.85
3280240	62	-59.03	-0.49
3352200	18	-103.03	-0.85
2864110	36	-85.03	-0.70
3272100	40	-81.03	-0.67
3072580	38	-83.03	-0.69
3112430	36	-85.03	-0.70
4020350	79	-42.03	-0.35
2764570	53	-68.03	-0.56
2012190	38	-83.03	-0.69
3144510	36	-85.03	-0.70
6281800	1	-120.03	-0.99
1040270	5	-116.03	-0.96
3144250	99	-22.03	-0.18
4036450	129	7.97	0.07
3424500	171	49.98	0.41
1436520	33	-88.03	-0.73
3648530	31	-90.03	-0.74
1744330	23	-98.03	-0.81
1484300	209	87.98	0.73
1488490	173	51.98	0.43
1100240	61	-60.03	-0.50
3804170	315	193.98	1.60
2444140	453	331.98	2.74
2128310	136	14.98	0.12
1964160	0	-121.03	-1.00
3412590	160	38.98	0.32
1088320	160	38.98	0.32
1928160	58	-63.03	-0.52
2924180	156	34.98	0.29
2468370	279	157.98	1.31
5482400	111	-10.03	-0.08
3205000	17	-104.03	-0.86
2776550	203	81.98	0.68
2944520	129	7.97	0.07
1168380	99	-22.03	-0.18

Logfile Swaps

Mean	StdDev	Median	Range
10.37	8.29	9	27

Session ID	Swaps	Deviation	Z-Score
1588430	24	13.63	1.64
9203300	20	9.63	1.16
3044340	12	1.63	0.20
1260580	6	-4.37	-0.53
3904260	6	-4.37	-0.53
3268210	14	3.63	0.44
4724100	0	-10.37	-1.25
1100230	27	16.63	2.01
3432480	27	16.63	2.01
2144250	13	2.63	0.32
2616130	3	-7.37	-0.89
1592150	13	2.63	0.32
2096480	4	-6.37	-0.77
3524360	8	-2.37	-0.29
3832220	27	16.63	2.01
3828330	16	5.63	0.68
3268500	12	1.63	0.20
3928900	0	-10.37	-1.25
2952200	6	-4.37	-0.53
1448130	10	-0.37	-0.04
1708180	4	-6.37	-0.77
2324500	3	-7.37	-0.89
3304540	4	-6.37	-0.77
2329000	10	-0.37	-0.04
3784900	6	-4.37	-0.53
3204120	18	7.63	0.92
1148330	0	-10.37	-1.25
3344410	6	-4.37	-0.53
3180160	12	1.63	0.20
2828370	0	-10.37	-1.25

SCML Swaps

Mean	**S**tdDev	**M**edian	**R**ange
12.85	17.25	6	73

Session ID	**S**waps	**D**eviation	**Z**-Score
2272570	20	7.15	0.41
5361700	54	41.15	2.39
2984520	73	60.15	3.49
2988580	42	29.15	1.69
2500430	6	-6.85	-0.40
3280240	8	-4.85	-0.28
3352200	3	-9.85	-0.57
2864110	8	-4.85	-0.28
3272100	3	-9.85	-0.57
3072580	4	-8.85	-0.51
3112430	6	-6.85	-0.40
4020350	4	-8.85	-0.51
2764570	8	-4.85	-0.28
2012190	3	-9.85	-0.57
3144510	6	-6.85	-0.40
6281800	0	-12.85	-0.74
1040270	0	-12.85	-0.74
3144250	7	-5.85	-0.34
4036450	42	29.15	1.69
3424500	42	29.15	1.69
1436520	6	-6.85	-0.40
3648530	6	-6.85	-0.40
1744330	6	-6.85	-0.40
1484300	47	34.15	1.98
1488490	30	17.15	0.99
1100240	6	-6.85	-0.40
3804170	2	-10.85	-0.63
2444140	0	-12.85	-0.74
2128310	11	-1.85	-0.11
1964160	0	-12.85	-0.74
3412590	4	-8.85	-0.51
1088320	10	-2.85	-0.17
1928160	6	-6.85	-0.40
2924180	3	-9.85	-0.57
2468370	12	-0.85	-0.05
5482400	9	-3.85	-0.22
3205000	0	-12.85	-0.74
2776550	4	-8.85	-0.51
2944520	7	-5.85	-0.34
1168380	6	-6.85	-0.40

Logfile Points

Mean	**S**tdDev	**M**edian	**R**ange
385.33	198.12	480	600

Session ID	**P**oints	**D**eviation	**Z**-Score
1588430	20	-365.33	-0.95
9203300	120	-265.33	-0.69
3044340	120	-265.33	-0.69
1260580	560	174.67	0.45
3904260	520	134.67	0.35
3268210	360	-25.33	-0.07
4724100	460	74.67	0.19
1100230	600	214.67	0.56
3432480	600	214.67	0.56
2144250	40	-345.33	-0.90
2616130	20	-365.33	-0.95
1592150	140	-245.33	-0.64
2096480	120	-265.33	-0.69
3524360	0	-385.33	-1.00
3832220	480	94.67	0.25
3828330	500	114.67	0.30
3268500	500	114.67	0.30
3928900	500	114.67	0.30
2952200	500	114.67	0.30
1448130	480	94.67	0.25
1708180	480	94.67	0.25
2324500	480	94.67	0.25
3304540	480	94.67	0.25
2329000	480	94.67	0.25
3784900	500	114.67	0.30
3204120	500	114.67	0.30
1148330	500	114.67	0.30
3344410	500	114.67	0.30
3180160	500	114.67	0.30
2828370	500	114.67	0.30

SCML Points

Mean	**S**tdDev	**M**edian	**R**ange
403.00	175.15	480	560

Session ID	Points	Deviation	Z-Score
2272570	240	-163.00	-0.93
5361700	20	-383.00	-2.19
2984520	40	-363.00	-2.07
2988580	200	-203.00	-1.16
2500430	500	97.00	0.55
3280240	500	97.00	0.55
3352200	480	77.00	0.44
2864110	500	97.00	0.55
3272100	480	77.00	0.44
3072580	480	77.00	0.44
3112430	500	97.00	0.55
4020350	500	97.00	0.55
2764570	500	97.00	0.55
2012190	500	97.00	0.55
3144510	500	97.00	0.55
6281800	500	97.00	0.55
1040270	500	97.00	0.55
3144250	380	-23.00	-0.13
4036450	40	-363.00	-2.07
3424500	140	-263.00	-1.50
1436520	540	137.00	0.78
3648530	560	157.00	0.90
1744330	560	157.00	0.90
1484300	0	-403.00	-2.30
1488490	200	-203.00	-1.16
1100240	380	-23.00	-0.13
3804170	480	77.00	0.44
2444140	480	77.00	0.44
2128310	480	77.00	0.44
1964160	500	97.00	0.55
3412590	0	-403.00	-2.30
1088320	500	97.00	0.55
1928160	500	97.00	0.55
2924180	480	77.00	0.44
2468370	480	77.00	0.44
5482400	500	97.00	0.55
3205000	500	97.00	0.55
2776550	480	77.00	0.44
2944520	480	77.00	0.44
1168380	520	117.00	0.67

Logfile Wizard Visits

Mean	**S**tdDev	**M**edian	**R**ange
2.67	2.34	2	7

Session ID	WizardVisits	Deviation	Z-Score
1588430	3	0.33	0.14
9203300	6	3.33	1.43
3044340	4	1.33	0.57
1260580	1	-1.67	-0.71
3904260	1	-1.67	-0.71
3268210	5	2.33	1.00
4724100	0	-2.67	-1.14
1100230	5	2.33	1.00
3432480	0	-2.67	-1.14
2144250	4	1.33	0.57
2616130	1	-1.67	-0.71
1592150	4	1.33	0.57
2096480	1	-1.67	-0.71
3524360	3	0.33	0.14
3832220	6	3.33	1.43
3828330	7	4.33	1.85
3268500	7	4.33	1.85
3928900	0	-2.67	-1.14
2952200	1	-1.67	-0.71
1448130	1	-1.67	-0.71
1708180	1	-1.67	-0.71
2324500	2	-0.67	-0.29
3304540	1	-1.67	-0.71
2329000	2	-0.67	-0.29
3784900	2	-0.67	-0.29
3204120	4	1.33	0.57
1148330	0	-2.67	-1.14
3344410	1	-1.67	-0.71
3180160	7	4.33	1.85
2828370	0	-2.67	-1.14

SCML Wizard Visits

Mean	**S**tdDev	**M**edian	**R**ange
2.65	3.64	1	14

Session ID	**W**izardVisits	**D**eviation	**Z**-Score
2272570	5	2.35	0.65
5361700	13	10.35	2.84
2984520	14	11.35	3.12
2988580	12	9.35	2.57
2500430	1	-1.65	-0.45
3280240	1	-1.65	-0.45
3352200	1	-1.65	-0.45
2864110	1	-1.65	-0.45
3272100	1	-1.65	-0.45
3072580	1	-1.65	-0.45
3112430	1	-1.65	-0.45
4020350	1	-1.65	-0.45
2764570	1	-1.65	-0.45
2012190	1	-1.65	-0.45
3144510	1	-1.65	-0.45
6281800	0	-2.65	-0.73
1040270	0	-2.65	-0.73
3144250	1	-1.65	-0.45
4036450	6	3.35	0.92
3424500	9	6.35	1.74
1436520	1	-1.65	-0.45
3648530	1	-1.65	-0.45
1744330	1	-1.65	-0.45
1484300	5	2.35	0.65
1488490	8	5.35	1.47
1100240	2	-0.65	-0.18
3804170	1	-1.65	-0.45
2444140	0	-2.65	-0.73
2128310	4	1.35	0.37
1964160	0	-2.65	-0.73
3412590	1	-1.65	-0.45
1088320	3	0.35	0.10
1928160	1	-1.65	-0.45
2924180	0	-2.65	-0.73
2468370	1	-1.65	-0.45
5482400	1	-1.65	-0.45
3205000	0	-2.65	-0.73
2776550	2	-0.65	-0.18
2944520	1	-1.65	-0.45
1168380	2	-0.65	-0.18

Logfile Dragon Visits

Mean	**S**tdDev	**M**edian	**V**ariance	**R**ange
2	1.98	1.5	3.93	8

Session ID	DragonVisits	Deviation	Z-Score
1588430	2	0.00	0.00
9203300	3	1.00	0.50
3044340	8	6.00	3.03
1260580	1	-1.00	-0.50
3904260	1	-1.00	-0.50
3268210	1	-1.00	-0.50
4724100	0	-2.00	-1.01
1100230	5	3.00	1.51
3432480	0	-2.00	-1.01
2144250	2	0.00	0.00
2616130	0	-2.00	-1.01
1592150	2	0.00	0.00
2096480	1	-1.00	-0.50
3524360	2	0.00	0.00
3832220	3	1.00	0.50
3828330	6	4.00	2.02
3268500	3	1.00	0.50
3928900	0	-2.00	-1.01
2952200	1	-1.00	-0.50
1448130	2	0.00	0.00
1708180	1	-1.00	-0.50
2324500	1	-1.00	-0.50
3304540	1	-1.00	-0.50
2329000	2	0.00	0.00
3784900	2	0.00	0.00
3204120	6	4.00	2.02
1148330	0	-2.00	-1.01
3344410	1	-1.00	-0.50
3180160	3	1.00	0.50
2828370	0	-2.00	-1.01

SCML Dragon Visits

Mean	**S**tdDev	**M**edian	**R**ange
1.75	2.59	1	12

Session ID	DragonVisits	Deviation	Z-Score
2272570	3	1.25	0.48
5361700	9	7.25	2.80
2984520	12	10.25	3.96
2988580	7	5.25	2.03
2500430	0	-1.75	-0.68
3280240	1	-0.75	-0.29
3352200	0	-1.75	-0.68
2864110	1	-0.75	-0.29
3272100	0	-1.75	-0.68
3072580	0	-1.75	-0.68
3112430	1	-0.75	-0.29
4020350	0	-1.75	-0.68
2764570	1	-0.75	-0.29
2012190	0	-1.75	-0.68
3144510	1	-0.75	-0.29
6281800	0	-1.75	-0.68
1040270	0	-1.75	-0.68
3144250	1	-0.75	-0.29
4036450	4	2.25	0.87
3424500	5	3.25	1.26
1436520	1	-0.75	-0.29
3648530	1	-0.75	-0.29
1744330	1	-0.75	-0.29
1484300	1	-0.75	-0.29
1488490	2	0.25	0.10
1100240	1	-0.75	-0.29
3804170	0	-1.75	-0.68
2444140	0	-1.75	-0.68
2128310	4	2.25	0.87
1964160	0	-1.75	-0.68
3412590	0	-1.75	-0.68
1088320	4	2.25	0.87
1928160	1	-0.75	-0.29
2924180	2	0.25	0.10
2468370	1	-0.75	-0.29
5482400	1	-0.75	-0.29
3205000	0	-1.75	-0.68
2776550	1	-0.75	-0.29
2944520	2	0.25	0.10
1168380	1	-0.75	-0.29

Logfile Doorway Attempts

Mean	**S**tdDev	**M**edian	**V**ariance	**R**ange
4.27	3.87	2	14.96	12

Session ID	DoorAttempts	Deviation	Z-Score
1588430	8	3.73	0.97
9203300	12	7.73	2.00
3044340	10	5.73	1.48
1260580	1	-3.27	-0.84
3904260	2	-2.27	-0.59
3268210	5	0.73	0.19
4724100	0	-4.27	-1.10
1100230	8	3.73	0.97
3432480	0	-4.27	-1.10
2144250	7	2.73	0.71
2616130	1	-3.27	-0.84
1592150	8	3.73	0.97
2096480	1	-3.27	-0.84
3524360	5	0.73	0.19
3832220	7	2.73	0.71
3828330	11	6.73	1.74
3268500	10	5.73	1.48
3928900	0	-4.27	-1.10
2952200	2	-2.27	-0.59
1448130	2	-2.27	-0.59
1708180	2	-2.27	-0.59
2324500	2	-2.27	-0.59
3304540	1	-3.27	-0.84
2329000	2	-2.27	-0.59
3784900	4	-0.27	-0.07
3204120	7	2.73	0.71
1148330	0	-4.27	-1.10
3344410	1	-3.27	-0.84
3180160	9	4.73	1.22
2828370	0	-4.27	-1.10

SCML Doorway Attempts

Mean	**S**tdDev	**M**edian	**R**ange
5.5	9.87	2	47

Session ID	DoorAttempts	Deviation	Z-Score
2272570	6	0.50	0.05
5361700	34	28.50	2.89
2984520	47	41.50	4.20
2988580	29	23.50	2.38
2500430	1	-4.50	-0.46
3280240	1	-4.50	-0.46
3352200	2	-3.50	-0.35
2864110	2	-3.50	-0.35
3272100	3	-2.50	-0.25
3072580	0	-5.50	-0.56
3112430	1	-4.50	-0.46
4020350	1	-4.50	-0.46
2764570	2	-3.50	-0.35
2012190	5	-0.50	-0.05
3144510	1	-4.50	-0.46
6281800	0	-5.50	-0.56
1040270	0	-5.50	-0.56
3144250	2	-3.50	-0.35
4036450	12	6.50	0.66
3424500	17	11.50	1.17
1436520	1	-4.50	-0.46
3648530	1	-4.50	-0.46
1744330	1	-4.50	-0.46
1484300	6	0.50	0.05
1488490	9	3.50	0.35
1100240	2	-3.50	-0.35
3804170	2	-3.50	-0.35
2444140	0	-5.50	-0.56
2128310	5	-0.50	-0.05
1964160	0	-5.50	-0.56
3412590	0	-5.50	-0.56
1088320	7	1.50	0.15
1928160	1	-4.50	-0.46
2924180	7	1.50	0.15
2468370	2	-3.50	-0.35
5482400	1	-4.50	-0.46
3205000	3	-2.50	-0.25
2776550	2	-3.50	-0.35
2944520	1	-4.50	-0.46
1168380	3	-2.50	-0.25

Appendix G

ACTION CATEGORIES

- atrans - Transfer of an abstract relationship(i.e. control, ownership)
- ptrans - Transfer of physical location of an object
- propel - The application of physical force to an object
- move - The movement of a body part of an animal
- grasp - The grasping of an object by an actor
- ingest - The taking of an object by an animal
- expel - The expulsion from the body of an animal into the world
- mtrans - The transfer of mental information between or within animals
- conc - The conceptualization or thinking about an idea by an animal
- mbuild - The construction of new information from old information
- attend - The action of directing a sense organ towards an object
- speak - The action of producing sounds from the mouth

REFERENCES

[1] ADAMS, D., *The Hitchhiker's Guide To The Galaxy.* New York, NY: Pocket Books, 1981.

[2] ALEXANDER, C., ISHAKAW, S., and SILVERSTEIN, M., *A Pattern Language.* New York: Oxford University Press, 1977.

[3] ANDERSON, J. S. and DURNEY, B., "Using Scenarios in Deficiency-driven Requirements Engineering," in *IEEE International Symposium on Requirements Engineering*, (San Diego, CA), pp. 134–141, IEEE Computer Society Press, 1993.

[4] ANDERSON, J., REDER, L., and SIMON, H., ""Situated Learning and Education," *Educational Researcher*, vol. 25.4, pp. 5–11, May 1996.

[5] ANDERSON, J. R., "Production Systems and the ACT-R Theory," in *Rules of the Mind*, pp. 1–14, Hillsdale, NJ: Erlbaum, 1993.

[6] ANTON, A. I., "Goal-Based Requirements Analysis," in *Proceedings of the Second IEEE International Conference on Requirements Engineering*, April 1996.

[7] AYYUB, B. M. and MCCUEN, R. H., *Numerical methods for engineers.* Upper Saddle River, NJ 07458, USA: Prentice-Hall, 1996.

[8] AYYUB, B. M. and MCCUEN, R. H., *Probability, Statistics, and Reliability for Engineers and Scientists.* Chapman & Hall/CRC, 2002.

[9] BAADER, F. and NARENDRAN, P., "Unification of Concept Terms in Description Logics," in *Proceedings of the International Workshop on Description Logics, DL'97*, pp. 34–38, LRI, Universitè PARIS-SUD, Cente d'Orsay, 1997.

[10] BARTLE, R., *Interactive Multi-User Computer Games.* Muse Ltd, British Telecom plc., 1990.

[11] BERGER, J. O., *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, 1993.

[12] BERLINER, B., "CVS II: Parallelizing Software Development," white paper, Prisma, Inc., Colorado Springs, CO, 1990.

[13] BÖHM, K., ABERER, K., NEUHOLD, E. J., and YANG, X., "Structured Document Storage and Refined Declarative and Navigational Access Mechanisms in HyperStorM," *The VLDB Journal*, vol. 6, pp. 296–311, Nov. 1997.

[14] BORHAUER, R., "Developing a Distributed Collaborative Battle Planning System - Integration of Legacy Display Systems," in *Proceedings of the ARL Federated Laboratory 4th Annual Symposium*, (College Park, MD), March 2000.

[15] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D., *Simple Object Access Protocol (SOAP) 1.1.* World Wide Web Consortium, May 2000. statut : « W3C Note, W3C Submission », http://www.w3.org/TR/SOAP/.

[16] Brooks, K., "Do Story Agents Use Rocking Chairs? The Theory and Implementation of One Model for Computational Narrative," in *Proceedings of the Fourth ACM Multimedia Conference (MULTIMEDIA'96)*, (New York, NY, USA), pp. 317–328, ACM Press, Nov. 1996.

[17] Bruckman, A. S., "The MediaMOO Project: Constructionism and Professional Community," *Convergence*, vol. 1, no. 1, 1995.

[18] Brutzman, D., Morse, K., Pullen, M., and Zyda, M., "Extensible modeling and simulation framework (xmsf): Challenges for web-based modeling and simulation," tech. rep., Naval Postgraduate School, Monterey, CA, 2002.

[19] Bush, V., "As We May Think," *The Atlantic Monthly*, vol. 176, pp. 101–108, July 1945.

[20] Campbell, J., *The Power Of Myth.* New York, NY: Doubleday, Inc., 1988.

[21] Carey, S., Kleiner, M., Heib, M., and Brown, R., "Standardizing Battle Management Language-A Vital Move Towards the Army Transformation," in *Proceedings of the 2001 Fall Simulations Interoperability Workshop*, (Orlando, FL), September 2001.

[22] Carr, F. and Myers, L., "Interoperability and Reuse through a Modeling and Simulation Common Operating Environment," in *Proceedings of the 2003 Spring Simulation Interoperability Workshop*, (Orlando, FL), April 2003.

[23] Carroll, J. M., "Editorial: Introduction to this Special Issue on "Scenario-Based System Development"," *Interacting with Computers*, vol. 13, no. 1, pp. 41–42, 2000.

[24] Carroll, J. M., ed., *Scenario-Based Design: Envisioning Work and Technology in System Development.* New York, NY: John Wiley & Sons, Inc., 1995.

[25] Carroll, J. M., "The Scenario Perspective on System Development," in *Scenario-Based Design: Envisioning Work and Technology in System Development* [24], pp. 1–15.

[26] Carroll, J. M., *Making Use: Scenario-based Design of Human-Computer Interactions.* Cambridge, MA.: MIT Press, 2000.

[27] Carroll, J. M., "Making Use: Scenarios and Scenario-Based Design," in *Proceedings of the ACM SIGCHI Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques (DIS-00)* (Boyarski, D. and Kellog, A. W., eds.), (N.Y.), pp. 4–4, ACM Press, Aug. 17–19 2000.

[28] Carson, G. S. and Henderson, L., "The CALS standard (military computing)," in *NCGA'90 Conference Proceedings*, (Fairfax, VA), pp. 570–598 (Vol. 1), National Computer Graphics Association, 1990. 3 volumes.

[29] Cavitt, D. B., Overstreet, C. M., and Maly, K. J., "Modeling and Distributed Simulation Techniques for Synthetic Training Environments," Technical Report TR_96_15, Old Dominion University, Apr. 25, 1996.

[30] Chen, P. P., ""The Entity-Relationship Model"," *ACM Trans. on Database Systems (TODS)*, vol. 1, pp. 9–36, 1976.

[31] Clemen, R. T., *Making Hard Decisions: An Introdution to Decision Analysis.* Belmont, CA: Wadsworth Publishing Company, second ed., 1996.

[32] Cline, M., "The Pros and Cons of Adopting and Applying Design Patterns in the Real World," *Communications of the ACM*, vol. 39, no. 10, pp. 37–39, 1996.

[33] Collins, A., Noonan, D., and Stark, E., *Complete Warrior (Dungeons Dragons Accessory).* Renton, WA: Wizards of the Coast, Inc., 2003.

[34] Columbia Encyclopedia, *The Columbia Encyclopedia.* Columbia University Press, sixth ed., 2004.

[35] Connolly, D., ed., *XML: Principles, Tools, and Techniques*, vol. 2(4) of *The World Wide Web Journal.* Cambridge, MA: O'Reilly & Associates, Inc., Winter 1997.

[36] Crittenden, R., *The Thames and Hudson Manual of Film Editing.* London: Thames and Hudson, Ltd, 1984.

[37] Crubezy, M. and Musen, M. A., "Ontologies in Support of Problem Solving," *Stanford Medical Informatics*, 2002.

[38] Dahmann, J., Fujimoto, R., and Weatherly, R., "The Department Of Defense High Level Architecture," Feb. 24 1999.

[39] Dautenhalm, K., "The Narrative Intelligence Hypothesis: In Search of the Transactional Format of Narratives in Humans and Other Social Animals," *CT 2001, LNAI 2117*, pp. 248–266, 2001.

[40] Dearden, A., Finlay, J., Allgar, E., and McManus, B., "Using Pattern Languages in Participatory Design," in *Proceedings of the Participatory Design Conference 2002*, (Malmo, Sweden), June 2002.

[41] Decker, D., "The American Screenplay," in *Anatomy of a Screenplay*, pp. 3–17, New York: The Screenwriters Group, 1998.

[42] Dällenbach, L., *The Mirror in the Text.* Chicago, Illinois: The University of Chicago Press, 1989.

[43] Dmytryk, E., *On Film Editing.* Boston, MA: Focal Press, 1984.

[44] Donnelly, C. and Stallman, R. M., *Bison Manual: Using the YACC-compatible Parser Generator.* Boston, MA, USA: GNU Press, 2002.

[45] D.Schmidt, Johnson, R., and Fayad, M., "Software Patterns," *Communications of the ACM*, vol. 39, no. 10, pp. 37–39, 1996.

[46] Dunnigan, J. F., *The Complete Wargames Handbook: How to Play, Design, & Find Them.* New York, NY: William Morrow and Company, 1992.

[47] Elmarisi, R. and Wu, Y.-C., "Conceptual Modeling for Customized XML Schemas," *Lecture Notes in Computer Science*, vol. 2503, pp. 429–443, 2002.

[48] Endsley, M., Pleban, R., and Matthews, M., "Measures of Platoon Leader Situation Awareness in Virtual Decision-Making Exercises," Tech. Rep. 1753, US Army Research Institute, Alexandria, VA, January 2000.

[49] Endsley, M. R. and Jones, D., "Designing to Support Situation Awareness in the Objective Force," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[50] Endsley, M. R. and Shattuck, L., "Situation Awareness Requirements for the Future Objective Force," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[51] Farquhar, A., Rice, J., and Fikes, R., "Tools For Assembling Modular Ontologies in Ontolingua," tech. rep., Jan. 16 2002.

[52] Fensel, D., Harmelen, F., Horrocks, I., and Klein, M., "The Relation Between Ontologies and XML Schemata," Feb. 04 2001.

[53] Fickas, S., Johnson, W. L., Karat, J., and Potts, C., "Using Scenarios to Elicit User Requirements," in *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, vol. 2 of *WORKSHOPS*, p. 467, 1994.

[54] Field, S., *The Foundations of Screenwriting.* New York, NY: Dell Publishing, 1984.

[55] Flanagan, M., Arble, F., Clanton, C., Marks, H., and Murray, J., "Interactive Narrative: Stepping into Our Own Stories," in *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems (Summary)*, vol. 2 of *Panels*, pp. 88–89, 1998.

[56] Foundation for Intelligent Physical Agents, "FIPA 98 Specification Part 1: Agent Management," Oct. 1998. version 1.0.

[57] Fowler, M. and Scott, K., *UML Distilled: Applying the Standard Modeling Object Language.* Object Technology Series, Addison-Wesley, 1997.

[58] Frensham, R., *Screenwriting.* Chicago, IL: NTC/Contemporary Publishing, 1996.

[59] Gall, M., Kutner, R., and Wesela, W., "The Proof and Illustration of the Central Limit Theorem by Brownian Numerical Experiments in Real Time within the Java Applet," *Lecture Notes in Computer Science*, vol. 3037, pp. 467–474, 2004.

[60] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns.* Reading, MA: Addison Wesley, 1995.

[61] Gehrke, M., Giese, H., Nickel, U., Niere, J., Tichy, M., Wadsack, J., and Zündorf, A., "Reporting about Industrial Strength Software Engineering Courses for Undergraduates," in *Proc. of the* 24^{th} *International Conference on Software Engineering (ICSE), Orlando, Florida,USA*, pp. 395–405, May 2002.

[62] Genesereth, M. R. and Fikes, R. E., "Knowledge Interchange Format, Version 3.0 Reference Manual," Tech. Rep. Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA, June 1992.

[63] Genter, D. and Markman, A., "Structure Mapping in Analogy and Similarity," *American Psychologist*, vol. 52.1, pp. 45–56, 1997.

[64] Gerdt, P., Kommers, P., Suhonen, J., and Sutinen, E., "StoryML: An XML Extension for Woven Stories," *Lecture Notes in Computer Science*, vol. 2363, pp. 893–??, 2002.

[65] Giere, R. N., ed., *Cognitive Models Of Science.* Minneapolis, MN: University Of Minnesota Press, 1992.

[66] Gonzalez, C., "ACT-R Implementation of an Instance-based Decision Making Theory," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[67] Gooding, D., "How Do Scientists Think? Capturing The Dynamics Of Conceptual Change In Science," in Giere [65], pp. 3–44.

[68] Gooding, D., "The Procedural Turn; or, Why Do Thought Experiments Work?," in Giere [65], pp. 45–76.

[69] Graham, I. S., *HTML 4.0 sourcebook: A Complete Guide to HTML 4.0 and HTML Extensions.* New York, NY, USA: Wiley, fourth ed., 1998.

[70] Groff, J. R. and Weinberg, P. N., *SQL: The Complete Reference.* Berkeley: McGraw Hill, 1999.

[71] Grose, T. J., Doney, G. C., and Brodsky, B., *Mastering XMI: Java programming with XMI, XML, and UML.* New York, NY, USA; London, UK; Sydney, Australia: John Wiley and Sons, 2002. Includes CD-ROM.

[72] Grosof, B. N. and Labrou, Y., "An Approach to Using XML and a Rule-Based Content Language with an Agent Communication Language," in *Issues in Agent Communication* (Dignum, F. and Greaves, M., eds.), pp. 96–117, Springer-Verlag: Heidelberg, Germany, 2000.

[73] Gruber, T. R., "A Translation Approach to Portable Ontology Specifications," Tech. Rep. KSL 92-71, Knowledge Systems Laboratory, Stanford University, April 1993.

[74] Gruber, T. R., "Ontolingua: A Mechanism to Support Portable Ontologies," tech. rep., Sept. 01 1994.

[75] Harel, I. and Papert, S., "Software Design as a Learning Environment," *Interactive Learning Environments*, pp. 1–32, 1990.

[76] Hayes, C. and Cunningham, P., "Shaping a CBR View with XML," *Lecture Notes in Computer Science*, vol. 1650, pp. 468–??, 1999.

[77] Hayes-Roth, B. and Hayes-Roth, F., "A Cognitive Model of Planning," in *Cognitive Science*, pp. 275–310, Ablex Publishing Company, 1979.

[78] Heffner, C. L., *Research Methods for Education, Psychology, and the Social Sciences*. Online E-textbook: AllPsych and Heffner Media Group, Inc., 2004.

[79] Henderson, C., "Model Execution in the OneSAF Objective System," in *Proceedings of the 2003 Spring Simulation Interoperability Workshop*, (Orlando, FL), April 2003.

[80] Hobbs, R., "Hyperscenarios: A Framework for Active Narrative," in *Proceedings of the 38th Annual ACM Southeast Conference*, (Clemson, SC), April 2000.

[81] Hobbs, R., "A Narrative Meta-Model Approach to Bridging M&S and C4I Applications," in *Proceedings of the 2003 Fall Simulations Interoperability Workshop*, (Orlando, FL), September 2003.

[82] Hobbs, R., "Using XML to Support Military Decision-making," in *Proceedings of the XML 2003 Convention & Exposition*, (Philadelphia, PA), Dec 2003.

[83] Hobbs, R. L., "Hypermedia Scenarios for Command & Control," in *Proceedings of the 13th Army Science Conference*, (Norfolk, VA), 1998.

[84] Hobbs, R. L., "An XML-Based Framework For Battle Planning Simulations," in *2000 Winter Simulations Conference*, (Orlando, FL), December 2000.

[85] Hobbs, R. L., "A Narrative-Based Cognitive Model For Computer-Generated Forces," in *Proceedings of the 10th Conference on Computer Generated Forces And Behavorial Representation*, (Norfolk, VA), May 2001. Abstract.

[86] Hobbs, R. L., "Sharing Stories: Using Narrative For Simulations Interoperability," in *Proceedings of the 2003 Spring Simulations Interoperability Workshop*, (Orlando, FL), Simulations Interoperability Standards Organization (SISO), March 2003.

[87] Hobbs, R. L. and Potts, C., "Towards a Framework for Hypermedia Scenarios," Technical Report GIT-CC-98-06, College of Computing, Georgia Institute of Technology, Atlanta, GA, Feb. 1998.

[88] Holmes, W. and Kogut, P., "XML + Semantics = DARPA Agent Markup Language," in *Lockheed Martin Joint Symposium 2001*, (Valley Forge, PA), June 2001.

[89] Holtzblatt, K. and Beyer, H., "Making Customer-Centered Design Work for Teams," *Communications of the ACM*, vol. 36, pp. 92–103, Oct. 1993.

[90] Horrocks, I., "DAML+OIL: A Reason-able Web Ontology Language," in *Proceedings of the Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27* (Jensen, C. S., Jeffery, K. G., Pokorný, J., Saltenis, S., Bertino, E., Böhm, K., and Jarke, M., eds.), vol. 2287 of *Lecture Notes in Computer Science*, Springer, 2002.

[91] Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F., "Reviewing the design of DAML+OIL: An ontology language for the semantic web," in *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-02)*, (Menlo Parc, CA, USA), pp. 792–797, AAAI Press, July 28– Aug. 1 2002.

[92] Houghton-Mifflin, *Webster's II Riverside Dictionary.* New York, NY: Berkeley Publishing Group, 1984.

[93] Houlihan, P., "Bookshelf: Design patterns for the enterprise: *Enterprise Modeling with UML: Designing Successful Software Through Business Analysis*," *IEEE Software*, vol. 18, pp. 106–107, Jan./Feb. 2001.

[94] Huang, J.-Y. and Deng, L. Y., "Modeling of the HLA-based simulation system," *Lecture Notes in Computer Science*, vol. 2195, pp. 367–??, 2001.

[95] International Organization for Standardization, *ISO/IEC 14977:1996: Information technology — Syntactic metalanguage — Extended BNF.* Geneva, Switzerland: International Organization for Standardization, 1996.

[96] Iscoe, N., Williams, G. B., and Arango, G., "Domain Modeling for Software Engineering," in *Proceedings of the 13th International Conference on Software Engineering*, pp. 340–343, May 1991.

[97] Jacky, J., *The Way of Z: Practical Programming with Formal Methods.* Cambridge University Press, 1997.

[98] James, H. S., "Correcting for Self-Selection Bias In Business Ethics Research," Working Paper 2004-08, Contracting and Organizations Research Institute (CORI), New York, NY, July 2004.

[99] Johnson-Laird, P. N., *Mental Models: Towards a Cognitive Science of Language, Inference, and Conciousness.* Cambridge: Cambrige University Press, 1983.

[100] Johnson-Laird, P. N. and Wason, P. C., eds., *Thinking: Readings In Cognitive Science.* Cambridge, MA: Cambridge University Press, 1997.

[101] Kalfoglou, Y. and Schorlemmer, M., "Information-Flow-Based Ontology Mapping," *CoopIS/DOA/ODBASE 2002*, vol. LNCS 2519, pp. 1132–1151, 2002.

[102] Kaste, R. and O'May, J., "From Simulation to Insights: Experiments in the Use of a Multi-criteria Viewer to Develop Understanding of the COA Space," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[103] Keefer, D. L., Kirkwood, C. W., and Corner, J. L., "Perspective On Decision Analysis Applications," *Decision Analysis*, vol. 1, pp. 4–22, Mar. 2004.

[104] Kerner, J., "Joint Technical Architecture: Impact on Department of Defense Programs," *Crosstalk, The Journal of Defense Software Engineering*, vol. Oct, 2001.

[105] Kesselman, J., Davis, M., Champion, M., and Pixley, T., "Document object model (DOM) level 2 specification," tech. rep., Mar. 07 2000.

[106] Kindler, E., Giese, H., Niere, J., Wadsack, J. P., Wendehals, L., Gehrke, M., Wagner, R., and Schfer, W., "Software Engineering Education," Feb. 11 2003.

[107] Klein, F., Seibel, A., and Giese, H., "Shuttle system case study," Case Study Description Version 1.0, University of Paderborn, Software Engineering Group, Oct. 2004.

[108] Kolodner, J., Owensby, J., and Guzdial, M., "Theory and Practice of Case Based Learning Aids," in *Handbook of Research for Educational Communications and Technology. 2nd edition*, pp. 829–861, Mahwah, NJ: Lawrence Erlbaum Associates, 2004.

[109] Kraaij, W., "TNO at CLEF-2001: Comparing Translation Resources," Aug. 20 2001.

[110] Kurtz, C., "StoryML: An XML Markup Language for Business Narrative," tech. rep., Knowledge Socialization Group. IBM T.J. Watson Research Center, Yorktown Heights, NY, 2000.

[111] Kyng, M., "Creating Contexts for Design," in Carroll [24], ch. 4, pp. 85–107. Aarhus University, Department of Computer Science.

[112] Landow, G. P., ed., *Hyper/Text/Theory.* Baltimore, MD: Johns Hopkins University Press, 1994.

[113] Laurel, B., *Computers as Theatre.* Reading, MA: Addison-Wesley Publishing Co., 1991.

[114] Laurel, B., Bates, J., Don, A., and Strickland, R., "Interface and Narrative Arts: Contributions from Narrative, Drama, and Film," in *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, Panels, pp. 381–383, 1991.

[115] Laurie, B. and Laurie, P., *Apache: The Definitive Guide.* 981 Chestnut Street, Newton, MA 02164, USA: O'Reilly & Associates, Inc., 1997.

[116] Law, A. M. and Kelton, D. W., *Simulation Modeling and Analysis.* New York, NY: McGraw-Hill, Inc., 1982.

[117] Lawrence, D. and Thomas, J. C., "Social dynamics of storytelling: Implications for story-base design," 1999.

[118] Liborg, N.-K., "A Study of Threats to Validity in Controlled Software Engineering Experiments." The University Of Oslo, 2004.

[119] Mack, R. L., "Discussion: Scenarios as Engines of Design," in Carroll [24], ch. 14, pp. 361–386. IBM T. J. Watson Research Center.

[120] Maeenpaeae, P., "Semantical BNF," *Lecture Notes in Computer Science*, vol. 1512, pp. 196–??, 1998.

[121] Mahemoff, M. J. and Johnston, L. J., "Pattern Languages for Usability: An Investigation into Alternative Approaches," in *Proceedings of the Asia-Pacific Conference on HCI 1998 (APCHI)*, (Los Alamitos, CA), July 1998.

[122] Mayfield, J., Labrou, Y., and Finin, T., "Evaluating KQML as an Agent Communication Language," in *Intelligent Agents II — Agent Theories, Architectures, and Languages (LNAI 1037)* (Wooldridge, M., Müller, J.-P., and Tambe, M., eds.), pp. 347–360, Springer-Verlag: Heidelberg, Germany, 1996.

[123] McGuinness, D. L., Fikes, R., Hendler, H., and Stein, L. A., "DAML+OIL: An ontology languages for the semantic web," *IEEE Intelligent Systems*, 2002.

[124] Mcnichols, K., Fadali, M. S., and Robinson, M., "Teaching Engineering to K-12 Students Using Role Playing Games," Aug. 30 2000.

[125] Mellor, S. J., Scott, K., Uhl, A., and Weise, D., *MDA Distilled: Principles of Model-Driven Architecture*. Boston: Addison-Wesley, 2004.

[126] *Merriam-Webster's Collegiate Dictionary*. Springfield, MA: Merriam-Webster, tenth ed., 1993.

[127] Merriam-Webster Online Dictionary, "Merriam-webster online dictionary," 2005.

[128] Milowski, A. and Richman, J., "Extensible stylesheet language (XSL) - version 1.0," tech. rep., World-Wide Web Consortium, Nov. 13 2001.

[129] Minsky, M., *The Society of Mind*. London: Heinemann, 1987.

[130] Mitchell, M. L. and Jolley, J. M., *Research Design Explained*. Pacific Grove, CA: Wadsworth, 5th ed., 2004.

[131] Mollaghasemi, M. and Pet-Edwards, J., *Making Multiple-Objective Decisions*. Los Alamitos, CA: IEEE Computer Society, 1997.

[132] Moore, M. and Potts, C., "Meso-Adaptation of Systems," Tech. Rep. BAA 00-20, DASADA Program, DARPA Technical Proposal, 2000.

[133] Murray, J. H., *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. New York, NY: Free Press, 1997.

[134] Nardi, B., *Information Ecologies: Using Technology With Heart*. Cambridge, MA: MIT Press, 1999.

[135] Nardi, B. A., "Some Reflections on Scenarios," in Carroll [24], ch. 15, pp. 387–399. Apple Computer, Advanced Technology Group.

[136] Nederland, A. A., Fensel, D., Harmelen, F. V., Horrocks, I., Erdmann, M., Klein, M., and Decker, S., "OIL in a Nutshell," July 23 2000.

[137] Nersessian, N., "The Procedural Turn; Or, Why Do Thought Experiments Work?," in Giere [65], pp. 45–76.

[138] Nichols, D. A. and Curtis, P., "MUDs Grow Up: Social Virtual Reality in the Real World," tech. rep., Xerox PARC, 1993.

[139] NIST, "Standard Generalized Markup Language (SGML)," tech. rep., National Institute for Standards and Technology (NIST), Gaithersburg, MD, September 1988.

[140] Nurcan, S., Grosz, G., and Souveyet, C., "Describing Business Processes with a Guided Use Case Approach," *Lecture Notes in Computer Science*, vol. 1413, pp. 339–??, 1998.

[141] Pearson, C. S., *The Hero Within: Six Archetypes We Live By*. New York, NY: Harper & Row, Publishers, Inc., 1989.

[142] Penna, G. D. and Intrigila, B., "An XML Definition Language for Software System Specification," in *Proceedings of the 6th World Multi-Conference on Systemics, Cybernetics,and Informatics*, (Orlando, FL), July 2002.

[143] Pohl, K. and Haumer, P., "Modeling Contextual Information about Scenarios," in *Proceedings REFSQ'97, 3rd Int. Workshop on RE*, (Barcelona, Spain), 1997.

[144] Potts, C., "Requirements Completeness, Enterprise Goals and Scenarios." Research Report, August 1994.

[145] Potts, C., "Using Schematic Scenarios to Understand User Needs," in *DIS95*, Scenarios, pp. 247–256, ACM, 1995.

[146] Potts, C., "Using Schematic Scenarios to Understand User Needs," in *Proceedings of the Symposium on Designing Interactive Systems: Processes, Practices, Methods and Techniques* (Olson, G. M. and Schuon, S., eds.), (New York), pp. 247–256, ACM Press, 23–25 Aug. 1995.

[147] Potts, C., "ScenIC: A Strategy for Inquiry-Driven Requirements Determination," in *RE'99: International Symposium on Requirements Engineering*, (Limerick, Ireland), IEEE Computer Society Press, 1999.

[148] Presley, A. and Liles, D. H., "Enterprise Modeling Within An Enterprise Engineering Framework," Sept. 10 1998.

[149] Propp, V., *Morphology Of The Folktale.* Austin, TX: University Of Texas Press, 1968.

[150] Restak, R. M., *The Mind.* New York, NY: Bantam Books, 1988.

[151] Roesler, A., Feil, M., and Woods, D. D., "Design is Sharing Stories About the Future," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[152] Rosenberg, J., "The Structure of Hypertext Activity," in *Proceedings of the 7th ACM Conference on Hypertext*, (New York), pp. 22–30, ACM Press, 16–20 Mar. 1996.

[153] Ross, K. G. and Klein, G., "The Recognition Planning Model: Application for the Objective Force Unit of Action," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[154] Rosson, M. B. and Carroll, J. M., *Usability Engineering: Scenario-Based Development of Human Computer Interaction.* Morgan Kaufmann Publishers, 2001.

[155] Rouse III, R., "Gaming and Graphics: Looking for Some Art Amidst the Technology," *Computer Graphics*, vol. 33, pp. 7–10, Feb. 1999.

[156] Rutledge, L., Alberink, M., Brussee, R., Pokraev, S., van Dieten, W., and Veenstra, M., "Finding the story: broader applicability of semantics and discourse for hypermedia generation," in *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HYPERTEXT-03)*, (New York), pp. 67–76, ACM Press, Aug. 26–30 2003.

[157] Sarkar, S. and Cleveland, C., "XML Based Document Transform Applied To Application Software Development Projects," Oct. 2001.

[158] Schank, R., Fano, A., Bell, B., and Menachem, J., "The Design of Goal Based Scenarios," *The Journal of the Learning Science*, vol. 3.4, pp. 305–345, 1993/1994.

[159] Schank, R. and Kass, A., "A Goal Based Scenario for High School Students," *Communications of the ACM*, vol. 39.4, pp. 28–29, April 1996.

[160] Schank, R. C., *Tell Me A Story: Narrative and Intelligence.* Evanston, Ill: Norwestern University Press, 1995.

[161] Schank, R. C. and Abelson, R. P., "Scripts, Plans and Knowledge," in *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, vol. 1, (Tbilisi, Georgia, USSR), pp. 151–157, IJCAI, Sept. 1975.

[162] Schengili-Roberts, K., *Core CSS: [Cascading Style Sheets].* Prentice Hall PTR core series, Upper Saddle River, NJ 07458, USA: Prentice-Hall PTR, 2000. Subtitle from cover.

[163] Scott, S., "Dueling Theories: Thought Experiments In Cognitive Science," in *CogSci2000*, (Philadelphia, PA), 22nd Cognitive Science Society Conference 2000, Aug. 2000.

[164] Segal, R. A., *Joseph Campbell: An Introduction.* New York, NY: Penguin Books USA, 1990.

[165] Shah, R. and Romine, J., *Playing MUDs on the Internet.* New York, NY: John Wiley & Sons, Inc, 1995.

[166] Shaker, S. and Gembecki, M., *The WarRoom Guide to Competitive Intelligence.* New York, NY: McGraw-Hill, Inc, 1999.

[167] Sharma, C. and Kunins, J., *VoiceXML: Strategies and Techniques for Effective Voice Application Development with VoiceXML 2.0.* Professional developer's guide series, New York, NY, USA: Wiley, 2002. Includes CD-ROM.

[168] Shimazu, H., "A Textual Case-Based Reasoning System Using XML on the World-Wide Web," *Lecture Notes in Computer Science*, vol. 1488, pp. 274–??, 1998.

[169] Simon, H. A., "What Is an 'Explanation' of Behavior?," *Psychological Science*, vol. 3, pp. 150–161, 1992.

[170] Simpson, J. E., *Just XML.* Upper Saddle River, NJ 07458, USA: Prentice-Hall PTR, 1999.

[171] Swick, R. R., "Resource Description Framework (RDF) Model and Syntax." World Wide Web Consortium, July 02 1998.

[172] Szilas, N., "Interactive Drama on the Computer: Beyond Linear Narrative," in *Proceedings of the AAAI 1999 Fall Symposium on Narrative Intelligence*, 1999.

[173] Taylor, A., "Perl for the DBA," *Sys Admin: The Journal for UNIX Systems Administrators*, vol. 8, pp. 20, 23–24, 26–27, 29–30, July 1999.

[174] Tolk, A., "Avoiding Another Green Elephant-A Proposal for the Next Generation HLA based on the Model-Driven Architecture(MDA)," in *Proceedings of the 2002 Fall Simulation Interoperability Workshop*, (Orlando, FL), September 2002.

[175] Tolk, A., "A Common Framework for Military M&S and C4I Systems," in *Proceedings of the 2003 Spring Simulations Interoperability Workshop*, (Orlando, FL), March-April 2003.

[176] Trochim, W., *The Research Methods Knowledge Base.* Cornell University: Atomic Dog Publishing, Inc., 2nd ed., 2001.

[177] US Army Command & General Staff College, Fort Leavenworth, KS, USA, *Battle Book*, July 1997.

[178] US Army Command & General Staff College, Fort Leavenworth, KS, USA, *Staff Organization and Operations*, May 1997.

[179] US Army Training and Doctrine Command, Fort Monroe, VA, *Operational Terms and Graphics*, Sept. 1997.

[180] US Department Of Defense, *High-Level Architecture Object Model Template Specification*, July 1998.

[181] US Department Of Defense, *High-Level Architecture Rules*, Apr. 1998.

[182] Wachowski, L., Wachowski, A., and Gibson, W., *The Matrix: The Shooting Script.* Newmarket Press, 2002.

[183] Warmer, J. and Kleppe, A., *The Object Constraint Language: Getting Your Models Ready for MDA.* Object Technology Series, Reading/MA: Addison-Wesley, Aug. 2003.

[184] Warwick, W. and Hayes, P., "Developing Computational Models of Naturalistic Decision-making: Methodologies and Perspectives," in *Proceedings of CTA Conference 2003 Advanced Decision Architectures*, (College Park, MD), April-May 2003.

[185] Watson, I., "A Case-Based Reasoning Application for Engineering Sales Support using Introspective Reasoning," Apr. 26 2000.

[186] Wieland, F., "The Threshold of Event Simultaneity," *TRANSACTIONS of The Society for Computer Simulation International*, vol. 16, no. 1, pp. 23–31, 1999.

[187] Wood, L., Hors, A. L., Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Nicol, G., Robie, J., Sutor, R., and Wilson (Eds), C., ""Document Object Model (DOM) Level 1 Specification Version 1.0"." W3C Recommendation, Oct. 1998. http://www.w3.org/TR/REC-DOM-Level-1/.

[188] World Wide Web Consortium, "Extensible markup language (XML) 1.0 (second edition)." W3C Recommendation, 2000. http://www.w3.org/TR/2000/WD-xml-2e-20000814.

[189] World Wide Web Consortium, "XML Schema Part 1: Structures, W3C Recommendation," May 2001. http://www.w3c.org/TR/xmlschema-1/.

[190] Yates, J. F. and Estin, P. A., "Decision making," in *A Companion To Cognitive Science* (Bechtel, W. and Graham, G., eds.), pp. 186–196, Malden, MA: Blackwell, 1998.

[191] Zsolt, B. and Ferenc, R., "Mining Design Patterns from C++ Source Code," in *Proceedings of the International Conference on Software Maintenance (ICSM '03)*, (Amsterdam, The Netherlands), September 2003.

VITA

REGINALD L. HOBBS is a research computer scientist in the Battlefield Information Processing Branch of the Army Research Laboratory. Mr. Hobbs has 20 years of experience in the computer industry, beginning on active duty as an information systems operator in the Air Force at NORAD headquarters. As a defense contractor, he worked as a programmer analyst, systems administrator, and computer facilities manager. Mr. Hobbs has a B.S. in Electronics from Chapman University and a Masters in Computer Science from the Georgia Institute of Technology. He is completing his PhD Candidacy in the area of Software Engineering at the College of Computing, Georgia Tech. He successfully defended his doctoral research as of June 2004. He has research interests and experience in the areas of software development methods, re-engineering, scenario-based design, XML for knowledge representation, and interactive narrative. Mr. Hobbs has participated on government and industry working groups on software engineering techniques, software process model descriptions, and software architecture transitions. Mr. Hobbs has extensive teaching experience at the undergraduate, graduate and continuing education level, teaching courses in software engineering, foundations of computer science, and information technology. He has been an instructor or adjunct professor at Clark-Atlanta University, Spelman College, Southern Polytechnic State University, and Georgia Tech Continuing Education. He and his wife, Lisa, have 5 children.

Education

B.S. Electronics, Chapman University, 1990

M.S. Computer Science, Georgia Institute of Technology, 1993

PhD Computer Science, Georgia Institute of Technology, 2005

Experience

Air Force Space Command, *Information Systems Operator* (Active Duty), 1984 - 1987

Army Research Institute, *System Administrator/Systems Programmer*, 1987 - 1989

Army Research Laboratory, *Computer Scientist*, 1990 - Present

North Carolina A&T, Industrial Engineering Dept., *Visiting Lecturer*, 1996-1998

Spelman College, CS Dept., *Adjunct Prof.,* 1998-2004

Southern Polytechnic State University, CS/IT Dept., *Adjunct Prof.,* 2000-2003

Georgia Tech Continuing Education Department, *Instructor*, 2000-2003

Areas of Teaching and Research Specialization

Software Engineering, Information Technology, Web Design & Development, Object-Oriented Analysis & Design (Java, UML, XML), Data Structures, Foundations of Computer Science, scenario-based design, interactive narrative, knowledge representation, cognitive science.

Publications

•R. Hobbs, **Show Me A Story: Capturing CGF Behavior Through Narrative** in Proceedings of the *Fourteenth Conference on Behavior Representation in Modeling and Simulation (BRIMS 05)*, Universal City, CA., May 16-19, 2005.

•R. Hobbs, **Hyperscenarios: Adding Domain Knowledge to Web-Enabled Simulations**, in Proceedings of the *2005 Spring Simulations Interoperability Workshop*, San Diego, CA., 3-8 April 2005. (In Press)

•R. Hobbs, **Using A Scenario Specification Language to Add Context to Design Patterns** in Proceedings of the *Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 04)*,Banff, Alberta, Canada., June 20-24 2004.

•R. Hobbs, **Using XML to Support Military Decision-making** in Proceedings of the *XML 2003 Convention & Exposition*, Philadelphia, PA., Dec 7-12 2003.

•R. Hobbs, **A Narrative Meta-Model Approach to Bridging M&S and C4I Applications**, in Proceedings of the *2003 Fall Simulations Interoperability Workshop*, Orlando, FL. September 2003.

•R. Hobbs**, Sharing Stories: Using Narrative for Simulations Interoperability,** in Proceedings of the *2003 Spring Simulations Interoperability Workshop,* Orlando, FL., 30 March - 4 April 2003. (In process)

•R. Hobbs, A Narrative-based Cognitive Model for Computer-Generated Forces• , in Proceedings of the *10th Conference on Computer Generated Forces and Behavioral Representation*, Norfolk, VA, 15-17 May 2001. (Abstract).

•R. Hobbs, **An XML-based Framework for Battle Planning Simulations•** , in Proceedings of the *2000 Winter Simulation Conference,* Orlando, FL., 10-13 December 2000.

•R. Hobbs, **Hypermedia Scenarios for Command & Control**, in Proceedings of the *21st Annual Army Science Conference: Science & Technology for the Army After Next*, Norfolk, VA, 15-17 June 1998.

•R. Hobbs, **Hypermedia Scenarios for Command & Control (Extended Abstract),** in *Summary Digest of the 21st Annual Army Science Conference*, Norfolk, VA, pp 147-148, 15-17 June 1998.

•R. Hobbs and C. Potts, **Towards a Framework for Hypermedia Scenarios**, College of Computing Technical Report GIT-CC-98-06, Georgia Institute of Technology, Atlanta, GA, 1998.

•R. Hobbs, G. Racine, **"Re-engineering Legacy Systems: A Case Study of Software Redevelopment and Business Process Improvement",** in *Proceedings of First Annual Software Engineering Techniques Workshop on Software Reengineering,* Software Engineering Institute (SEI), Pittsburgh, PA, 3-5 May 1994.

•R. Hobbs, **"COBOL-to-Ada Transition: A System Re-engineering Case Study",** *Computer-Aided Software Engineering - Issues and Trends for the 1990s and Beyond*, Idea Publishing Group, Harrisburg, PA, 1993.

•R. Hobbs, J. Mitchell, G. Racine, **"Ada Transition Research Project (Phase II): Final Report",** Army Institute for Research in Management Information, Communications, and Computer Science (AIRMICS) Report ASQB-GI-92-004, AIRMICS, Atlanta, 1992.

•R. Hobbs, J. Mitchell, G. Racine, **"Re-engineering Old Production Systems: A Case**

Study of Systems Re-development and Evaluation of Success", in Emerging Information Technologies for Competitive Advantage and Economic Development, *Proceedings of 1992 Information Resources Management Association (IRMA) International Conference,* Charleston, SC, pp 29-37, 24-27 May 1992.

•G. E. Racine, R. Hobbs, R. Wassmuth, **"Ada Transition Research Project (A Software Re-engineering Effort)",** in *Proceedings of the Tenth Annual National Conference on Ada Technology (ANCOAT `92),* Arlington, VA, pp 192-201, 24-28 February 1992.

•R. Hobbs, J. Nealon, R. Wassmuth, **"Ada Transition Research Project (Phase I): Final Report",** Army Institute for Research in Management Information, Communications, and Computer Science (AIRMICS) Report ASQB-GI-91-005, AIRMICS, Atlanta, 1991.

•R. Hobbs, **"System Reengineering Executive Summary",** Army Institute for Research in Management Information, Communications, and Computer Science (AIRMICS) Report ASQB-GI-92-003, AIRMICS, Atlanta, 1991.