A WHITEBOARD-BASED COMPUTATIONAL FRAMEWORK FOR DYNAMIC TEAM COORDINATION

by Jae Woong Hwang

B.S. in Computer Science, 1992, Virginia Tech
M.S. in Computer Science, 1995, Virginia Tech

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

January 31, 2010

Dissertation directed by

Shmuel Rotenstreich
Associate Professor of Computer Science

The School of Engineering and Applied Science of The George Washington University certifies that Jae Woong Hwang has passed the Final Examination for the degree of Doctor of Philosophy as of November 18, 2009. This is the final and approved form of the dissertation.

A WHITEBOARD-BASED COMPUTATIONAL FRAMEWORK FOR DYNAMIC TEAM COORDINATION

Jae Woong Hwang

Dissertation Research Committee:

Shmuel Rotenstreich, Associate Professor of Computer Science
Dissertation Director

Bhagirath Narahari, Professor of Computer Science
Committee Member

Poorvi Vora, Associate Professor of Computer Science
Committee Member

Xiuzhen Cheng, Associate Professor of Computer Science
Committee Member

Nils Olson, Assistant Professor of Organizational Sciences & Communication
Committee Member

© Copyright 2009 by Jae Woong Hwang
All rights reserved

Dedication

To My Parents

Ik Nam Hwang and Shin Ja Hwang

Acknowledgments

At first, I would like to thank my research advisor, Dr. Shamuel Rotenstreich, for his excellent guidance throughout the research. His invaluable comments always helped me to progress toward the next steps. I would like to extend many thanks to the dissertation committee, Dr. Bhagirath Narahari, Dr. Poorvi Vora, Dr. Xiuzhen Cheng, and Dr. Nils Olson, for their insightful suggestions to improve the dissertation in many ways.

I would like to thank all colleagues in the research group. The early conversations with Dr. George Dimitoglou allowed me to draw an overall roadmap for the dissertation. The research meetings continuously motivated me to progress. I owe much gratitude to all colleagues: Lewis Berman, Horatious Tanyi, Amelia Estwick, Syed Omair Hasan, Dave Christian, Rey-er Lee, Ray D. Renner, George Blankenship, Bradley Taylor, Farhad Heydari, Aleksandar Stefanovski, and Carroll Vidrine.

I would also like to thank Dr. Jang Koon Shin, the formal Department Head of Computer Science at Korea Military Academy. An opportunity to work in an academic environment was an invaluable experience prior to this dissertation work. I am also much grateful to Dr. Albert Hoang, who has been a friend and a fellow throughout my doctoral study. The conversations with him was always fun and inspired me of new perspectives. I wish the best for his second Ph.D.

Finally but most importantly, I would like to thank all of my family. Without their unconditional love and care, I doubt if I would ever been able to complete this work. This dissertation is dedicated to my parents who have been always supportive regardless of any circumstances in life.

Abstract

A WHITEBOARD-BASED COMPUTATIONAL FRAMEWORK FOR DYNAMIC TEAM COORDINATION

This dissertation presents a whiteboard-based computational framework to support dynamic team coordination in a distributed environment. Coordination requires awareness of situations and policies. Policies are rules governing team situations in an organization where situations are contextual information. A whiteboard supports awareness by full information sharing. The framework provides a whiteboard-based coordination model that relies on awareness of team situations and organizational policies. It consists of principle components that address the keys issues such as representation, communication, log transfer syntax, and negotiation. A whiteboard component manages contexts and rules, represented with Semantic Web technologies. It provides flexible yet interoperable communications to cope with diverse interactions. A simple line-based scripting language supports the transfer syntax of fully-structured event log messages. A policy-based negotiation protocol allows a team to find a collaboration partner, adaptively coping with dynamic situations and policies.

Table of Contents

Dedication iv

Acknowledgments v

Abstract vi

Table of Contents vii

List of Figures xi

Chapter 1 - Introduction 1

1.1 Problem Statement 2

1.2 Outline 4

Chapter 2 - Literature Review 5

2.1 Coordination 5

2.2 Whiteboard 6

2.2.1 Whiteboard in a Human Organization 6

2.2.2 Whiteboard as a Computational Model 7

2.3 Representation 10

2.3.1 What is Context? 11

2.3.2 Context Model 11

2.3.3 Semantic Web 13

2.4 Communication and Architecture 14

2.5 Transfer Syntax 17

2.6 Negotiation 19

Chapter 3 - Whiteboard Component 21

3.1 Ontology-based Data Representation 21
3.2 Policies and Queries 24
3.3 Whiteboard Types 25
3.4 Whiteboard Operations 26
3.5 Implementation Layer 28
Chapter 4 - Communication and Architecture 29
4.1 Computing Environment 29
4.2 Problem Domain: Logical Layout 30
4.3 Agent Types 31
4.4 Collective System Behavior 32
4.5 Communication Styles 32
4.5.1 Web Services 33
4.5.2 REST 33
4.5.3 Pipeline 34
4.6 Implementation Architecture 35
4.7 Logical Layers of System Components 36
Chapter 5 - Context-aware Event Log Language 37
5.1 5W1H-based Event Log Structure 37
5.2 Comparison 39
5.2.1 Traditional Logs and CELL 39
5.2.2 XML and CELL 41
5.3 CELL Language Specification 43
5.3.1 Object Specification 43

5.3.2 Property Specification 44
5.3.3 Named Object 45
5.3.4 Unnamed Object 46
5.3.5 Nested Object 46
5.3.6 Rules and Queries 46
5.3.7 Coquery 48
5.3.8 References: Whiteboard Service, Ontology, and Namespace 48
5.3.9 Special Features: Line Tag, Whiteboard Alias, and Block 49
5.3.10 Comments and Multi-line specification 50
5.3.11 Utility Commands 50
5.4 Translation and Whiteboard Integration 51
5.5 CELL EBNF 52
Chapter 6 - Situation/Policy-Aware Negotiation 54
6.1. Simple Coordination Example 55
6.1.1 Transport - Team Working Environment 55
6.1.2 Scenario – Demanding Collaboration 56
6.2. An Adaptive Policy-based Negotiation Protocol 58
6.2.1 Finding Peer Teams 58
6.2.2 Candidate Selection 59
6.2.3 Expand and Retry 60
6.2.4 Decision Making 61
Chapter 7 - Testing and Evaluation 65
7.1 Testing Environment and Approach 65

7.2 Functional Testing: Whiteboard Operations and Services 67
7.2.1 Procedure 67
7.2.2 Test Cases for Core Whiteboard Operations 67
7.2.3 Test Cases for Service-level Operations 70
7.3 Functional Testing: CELL Scripts 70
7.3.1 Procedure 70
7.3.2 Test Cases Using CELL Scripts 71
7.3.3 Whiteboard Content Visualizations 72
7.4 Performance Analysis 76
7.4.1 Procedure 77
7.4.2 Result of Performance Measures 78
7.4.3 Result of Variant Cluster Size 80
7.5 Discussion 82
Chapter 8 - Conclusions and Future Work 84
8.1 Conclusions 84
8.2 Future Work 85
8.2.1 Improvements 86
8.2.2 Integration and Interoperability 87
References 88

List of Figures

Figure 1-1 Key Issues 2

Figure 2-1 Whiteboard Architecture 8

Figure 3-1 Two-Stage Definition of Ontologies 22

Figure 3-2 Relationships in Ont-Team Ontology 23

Figure 3-3 Whiteboard Contents - Ont-Transport Ontology Instances 23

Figure 3-4 Whiteboard Implementation Layer 28

Figure 4-1 Agents 30

Figure 4-2 Logical Layout of Agents and Teams 30

Figure 4-3 Agent Types 31

Figure 4-4 Communication Styles 33

Figure 4-5 Pipeline - Log Streams 34

Figure 4-6 Agent Components 35

Figure 4-7 System Component Layers 36

Figure 5-1 Data Structure and Ontology 38

Figure 5-2 Event Log in OWL/XML 42

Figure 5-3 Event Log in CELL 43

Figure 5-4 CELL EBNF 53

Figure 6-1 Transport Example 55

Figure 6-2 Negotiation Protocol 57

Figure 6-3 Expanding Peer teams 59

Figure 6-4 Distributed Context Sharing with Multiple Whiteboards 61

Figure 7-1 Test Case for Objects and Properties 68
Figure 7-2 Test Case for Building Blocks 68
Figure 7-3 Test Case for Rules and Queries 69
Figure 7-4 Test Case for coquery 69
Figure 7-5 CELL script for Object Creation 71
Figure 7-6 CELL Script for Event Log 71
Figure 7-7 CELL Script for Rules and Queries 72
Figure 7-8 CELL Script for coquery 72
Figure 7-9 Sample A - Whiteboard Ontology Visualization by Protégé/Owlviz 73
Figure 7-10 Sample B - Whiteboard Ontology Visualization by Protégé/Ontoviz 73
Figure 7-11 Sample C - Whiteboard Ontology Visualization by Protégé/Ontoviz 74
Figure 7-12 Sample D - Whiteboard Contents Visualization by Protégé/Jambalaya 75
Figure 7-13 Sample E - Whiteboard Content Visualization by Protégé/Jambalaya 75
Figure 7-14 Sample F - Whiteboard Content Visualization by Protégé/Jambalaya 76
Figure 7-15 Performance Measures - Every 5000 Objects 79
Figure 7-16 Performance Measures - Every 1000 Objects 79
Figure 7-17 T_CREATE Performance 80
Figure 7-18 Performance of Different Cluster Sizes 81

Chapter 1 - Introduction

Team coordination in a distributed computing environment is dynamic and complex. Teams have to be flexible and adaptive to the constantly changing conditions. Coordination is necessary to cope with limited resources. A team with insufficient resources may decide to collaborate with other teams. Choosing a partner requires a negotiation protocol to cope with team situations and organizational policies. Policies are the rules governing team situations in an organization. Situations are characterized by context information [1], describing person, place, or object that is considered relevant to interactions.

Team coordination requires awareness of situations and policies. Awareness can be achieved by full information sharing. A whiteboard [2, 3, 4] is a communication medium that provides information sharing among teams. It supports coordination by allowing teams to achieve awareness of dynamic situations. In a human organization, it may contain a physical collection of messages, notes, or other artifacts to be shared. For a computational model, a whiteboard can be a service-based component that manages context logs and rules.

There are many team collaboration approaches and only a few of them are policy driven. Furthermore, none of the existing policy driven models offers awareness and flexibility under changing conditions. In addition, sharing information and context among teams is not fully supported [5]. The proposed team coordination model offers full context sharing.

This dissertation presents a whiteboard-based computational framework to support team coordination. The framework provides a computational model for dynamic teams to achieve awareness of situations and policies in a distributed environment. It addresses the key issues for building a framework in terms of representation, communication, log transfer syntax, and negotiation.

Figure 1-1 visualizes a team working environment with a whiteboard. A support for communications within a team and among teams is necessary. Log transfer syntax specifies a log format deliverable from a team member to a team leader. For collaboration, a negotiation is required among teams.

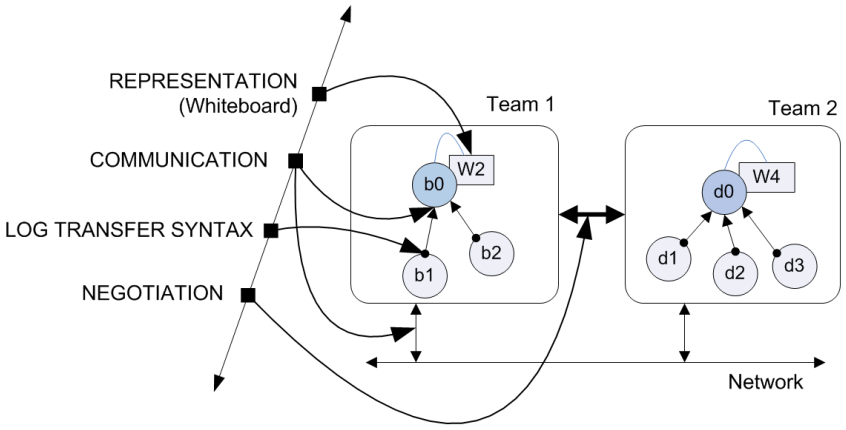

Figure 1-1 Key Issues

1.1 Problem Statement

This dissertation is to develop a whiteboard-based computational framework for dynamic team coordination in a distributed environment. The framework supports coordination that relies on awareness of situations and policies with:

- A whiteboard representation for contexts and rules
- A flexible communication architecture
- A line-based log transfer syntax
- A policy-aware adaptive negotiation protocol

The whiteboard-based framework consists of four principle components to address representation, communication, log transfer syntax, and negotiation for team coordination. The major works in the research are the developments of those components for the framework.

First, a whiteboard component is constructed to manage contexts and rules. It facilitates the Semantic Web [6, 7, 8] technologies to provide a rich representational framework [9] using ontologies and rules. The component defines the whiteboard representations for contexts, policies, and queries. It also specifies whiteboard types and operations to manage the whiteboard contents.

Second, a whiteboard agent is created to provide flexible yet interoperable communications to cope with diverse interactions. It utilizes a hybrid approach with multiple protocols and architecture styles including Web Services [10, 11, 12], Representational State Transfer (REST) [13, 14, 15, 16], and pipeline [17, 18]. It supports the RPC-style invocations to whiteboard operations, the sharing of distributed contexts from multiple whiteboards, and the delivery of log contexts as data streams.

Third, Context-aware Event Log Language (CELL) is designed to specify the transfer syntax of fully-structured event logs. It is a simple scripting language that can represent

semantic logs as data lines. A translator is required to interpret log contexts and store them into a whiteboard.

Forth, Situation/Policy-Aware Negotiation (SPAN) protocol is specified to support a policy-based negotiation for team collaboration. The protocol finds a collaboration partner, adaptively coping with policies that govern team situations. A simple team coordination scenario is created to describe the details of the protocol logics.

As a proof of concept, a prototype is implemented and tested. It includes core features of components that are evaluated in terms of functional testing to verify the correctness of operations and a performance analysis to examine capabilities and potential issues.

1.2 Outline

The remainder of this dissertation is organized as follows. Chapter 2 provides a literature review in the related area. Chapter 3 introduces the whiteboard component in terms of representations, operations, and implementations. Chapter 4 discusses the communication and architecture in the framework. Chapter 5 and 6 describe the CELL language and the SPAN protocol, respectively. Chapter 7 discusses testing and evaluations of a prototype. Chapter 8 presents the conclusions and future work.

Chapter 2 - Literature Review

This chapter provides a survey on related works. Developing a framework is a complex task that requires a background research on several areas and underlying technologies in order to build a ground model. Among many related approaches and technologies, the review is grouped into a few categories to address the key issues: coordination, whiteboard, representation, communication, transfer syntax, and negotiation.

2.1 Coordination

Marlone et al. [19, 20] define "coordination is managing dependencies between activities," where common dependencies occur due to shared resources, produce/consumer relationships, simultaneity constraints, and task/subtasks. It explains that without interdependencies, no coordination is required.

Espinosa et al. [21, 22] explains that team coordination is also accomplished by an implicit mechanism, relying on shared knowledge that is necessary to coordinate task activities and operate as a consistent unit. It is characterized with shared mental models that are developed over time through experience. Shared mental models are made of two principal components: a common or consistent model of the tactical situation among team members and a set of mutual mental models about the other team-member functions [23, 24, 25]. It allows team members to preempt the actions and needs of another, reducing communication and explicit coordination overheads.

Xiao et al. [4] describes that computation and communication technologies may improve team coordination that relies on awareness of dynamic tasks and team situations. The case studies of teams in the trauma center [26, 27] report a display-based whiteboard has evolved into a key component for supporting collaboration in a rapid paced and highly dynamic working environment. It explains that the whiteboard may support activities such as negotiation of scheduling, joint planning, and inter-personal communication.

2.2 Whiteboard

This section reviews different approaches that utilize a whiteboard in the area. A whiteboard in a human organization is a medium that supports information sharing among people. For a computing environment, the principle idea is similar to a blackboard, but the focus appears to be on an indirection communication although several different approaches exist to cope with a whiteboard. The rest of this section summaries known approaches for a whiteboard.

2.2.1 Whiteboard in a Human Organization

A whiteboard [2, 3] is a communication medium that allows people to share information. People take notes, outline to-do lists, and organize any artifacts in an office working environment. It is one of most common tools supporting interactions, collaborations, and discussions with other colleagues. Mynatt et al. [2, 28] describes that a whiteboard acts as a working area and repository to support tasks of which the

emphasis is on understanding ideas and concepts. Whiteboard content is heavily context-dependent such that it presents users to remember why and when it was written.

Igarashi et al. [3, 28] presents the software architecture for a pen-based electronic whiteboard system, called Flatland. It is a type of a window system for pen-based applications that is designed to support various activities on personal office whiteboards with ease of use. It provides a mechanism for managing the history on the board such that a user's experience with the whiteboard is captured and available for retrieval. Shae et al. [29] presents an immersive whiteboard system that allows the communication among distributed users. The whiteboard in their system is "both a physical platform for the input/output interfaces and a virtual screen for sharing common multimedia" where the virtual world with avatars is projected based on a user's position and viewpoint. It allows the user to physically write on the virtual whiteboard using a pen interface, and the stroke data are delivered by streaming. Zhang et al. [30] describes a technique that transforms the contents on a physical whiteboard into an electronic document. It captures the contents into images with a digital camera, and performs a serious of image processing procedures to transform it into a rectangular bitmap and a vectorized representation (inks) that can be used in a TablePC.

2.2.2 Whiteboard as a Computational Model

A whiteboards as a computational model shows similar characteristics of a blackboard. In a blackboard system, multiple software modules work cooperatively to solve a problem by communicating through a shared data repository, or a blackboard [31,

32, 33]. It is known to be a useful construct for facilitating interactions among heterogeneous components with complex run-time behaviors [34, 35, 36]. The principal idea behind a blackboard is modularity where "a problem space is broken up into problem areas, each of which has its own component or "expert" working on it, communicating with experts in other areas via the blackboard, posting information to it and reading information from it [37]."

Boitet et al. [38] introduces a whiteboard architecture as a way to integrate the heterogeneous components in the distributed Natural Language Processing (NLP) systems. A whiteboard is described as an intermediate architecture between sequential [39, 40] and blackboard [41, 42, 43] architectures where components are integrated under a coordinator that maintains the input and output data structures of each components in a whiteboard. Each component process is encapsulated in a manager that communicates with the coordinator. Figure 2-1 shows the whiteboard architecture in the paper.

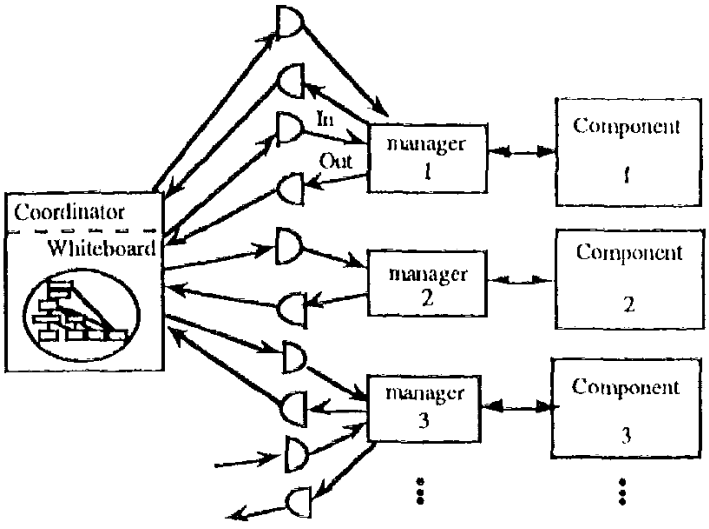

Figure 2-1 Whiteboard Architecture [38]

It explains that a sequential approach offers an easy solution for the integration, but leads to loss of information and lack of robustness where partial processing is not allowed. The blackboard architecture exposes the complexity of controlling concurrent accesses, communication overheads, and efficiency problems due to the direct accesses to the blackboard by components. It is considered to be a whiteboard architecture if the system uses global data structures and records that are accessed only by a coordinator and hidden from the components.

Cahill et al. [44, 45] presents a whiteboard as a reference architecture for Natural Language Generation (NLG) systems. It describes that the whiteboard divides processing into knowledge-centric functional modules where they can communicate through a single repository of data. A whiteboard can facilitate the sharing and reuse of processing resources such as modules that are the codes to perform specific linguistic tasks as well as data resources including inputs, lexicons, and grammars. It defines a whiteboard as a "cumulative typed relational blackboard." The whiteboard is typed and relational because its data are represented by objects, types, and relations. It is similar to a blackboard because of its shared control architecture and data store among modules. However, its data are not destructive as in the blackboard. The data stored in a whiteboard are not changed or removed, but its structure is built incrementally by making successive copies that are linked by revises links.

Neophytou et al. [46] describes a data dissemination infrastructure for collaborative applications in a disaster management environment where the mass delivery of messages to a wide variety of sources is demanded and the resources from several sources need to get together. It proposes a combined approach using Peer-to-Peer (P2P) and whiteboard

architectures for reliable delivery of data. It explains that the whiteboard architecture is intended to facilitate coordination of cooperating modules where it can act as a wrapper for legacy systems and dynamically adapt to changing network topology on demand. The whiteboard may be an intermediary for indirect connections by modules. As such, the ordering of operations based on data flow and module participation can be dynamically determined at runtime that may enhance reliable data dissemination.

Thorisson et al. [37] introduces a whiteboard as a modern descendant of a blackboard that is not built specifically for supporting reasoning and expert systems, but is somewhat closer to network routing technologies. The principal idea behind the blackboard is modularity that allows the integration of heterogeneous components requiring fine-grained interactions through the blackboard. It describes that the whiteboard extends the blackboard with the message passing capability that can better support the multi-module integration to build a larger complex system. The proposed whiteboard system supports the communications based on publish-subscribe and querying. The messages and data streams are represented with dot-delimited lists where each delimited segment is ontologically defined. The paper explains that it is simple but powerful message semantics.

2.3 Representation

This section discusses the issues related to representations of situational information or contexts. It reviews the commonly-referenced definition of context as well as different approaches for modeling contexts.

2.3.1 What is Context?

For humans, context is implicit situational information [1] that helps them to share common understanding of how the world works. For a computational model, Schilit et al. [47, 48] first discusses context-aware computing as software that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time." The important aspects of context are "where you are, who you are with, and what resources are nearby". Pascoe [49] describes that context is the subset of physical and conceptual states of interest to a particular entity. Dey [1] claims that context defined by prior works is not feasible to apply in practice for an application development. It explains that it only gives the synonyms for context such as the environment or situation, or it is too specific by enumerating a fixed set of aspects. Context is about any situations relevant to an application and users, and it may vary from situation to situation. It gives an operational definition of context such that "context is any information that can be used to characterize the situation of entity" where an entity is a person, place, or object that is considered relevant to interactions.

2.3.2 Context Model

Strang et al. [50] provides a survey on context modeling approaches. A context model with key-value pairs [47, 51, 52] uses a simple data structure where context information such as a location is represented as a key and its associated value is utilized for an application. A markup-based model [53, 54] is commonly represented with

profiles that form a hierarchical data structure with tags using Extensible Markup Language (XML) [55] or Standard Generalized Markup Language (SGML) [57]. A graphical model such as Entity-Relationship (ER) Model [58], Unified Modeling Language (UML) [59, 60], and Object Role Modeling (ORM) [61, 62, 63, 64] provides the visual description of the context structure that is typically transformed to codes or other model. An ER model is frequently used for database systems [65, 66, 67]. A context model with an Object-Oriented Paradigm (OOP) [68] may simplify the knowledge representation for a complex system by facilitating features such as information hiding, data abstraction, encapsulation, inheritance, and reusability [69]. A logic-based model [70, 71, 72, 73, 74] shows a high degree of formality that often relies on rules for validation and reasoning.

Strang et al. [50] reports ontologies as a promising technology for modeling contexts. An ontology is a collection of concepts and their relationships that provides an abstract view of an application domain [75, 76, 77]. Krummenacher et al. [78] describes that ontologies are a widely-accepted tool [79, 80, 81, 82, 83] for modeling context information where real-life entities can be represented in machine-readable constructs. It allows the shared understanding of contextual knowledge and information that can be interpreted and evaluates by logic-based reasoning [84]. Strang et al. [50] describes that ontologies encompass the representational strength of OOP due to the similarities between ontologies and objects where concepts and individuals in ontologies are analogous to classes and instances in OOP. The ontology-based modeling further allows a high-level of formality where the precise definition of terms and relationships can be specified allowing the shared interpretation of the exchanged data.

Wang et al. [84, 52] proposes an ontology, CONON, for modeling context in a pervasive computing environment where applications and services must be aware of and adapt to dynamically-changing contextual information. It is based on Web Ontology Language (OWL) [85, 87]. CONON provides two sets of ontologies. An upper ontology captures general concepts about basic context such as location, person, and activity and an extended ontology to represent the domain-specific knowledge. Chen et al. [86] proposes Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) as a step towards the standardization of a shared ontology. Similar to CONON, it facilitates OWL to supports two sets of ontologies, SOUPA Core and SOUPA Extension, in which the SOUPA Core addresses the required generic concepts for contextual information including time, space, events, and user. It reports that the use of ontologies is a key requirement for realizing the vision of pervasive computing that requires services and information in an "anywhere and anytime" fashion.

2.3.3 Semantic Web

Semantic Web is a vision of the next generation World Wide Web [6, 7, 8] that supports a new ontology-based knowledge representation framework to cope with the unstructured digital information on Web. It helps machines to better understand information with the semantically-rich and descriptive representation of information on any resource [88]. With its development focusing on ontology-based technologies and tools, several ontology languages are introduced such as DAML + OIL [89], Resource Description Framework (RDF) [90], Notation 3 (N3) [91], and Web Ontology Language

(OWL) [85, 87]. It is currently being converged to OWL as a W3C standard [92]. Semantic Web Rule Langue (SWRL) [93] is a proposed rule language for Semantic Web that is based on the combination of OWL and Rule Markup Language (RuleML) [94]. Semantic Query-enhanced Web Rule Language (SQWRL) [95] is a SWRL-based query language that is developed as a part of the Protégé project [96].

Because Semantic Web relies on ontology-based technologies, it inherits all benefits of ontology-based context modeling. Additionally, it may support a policy-aware representation using its associated rule language, SWRL. Weitzner et al. [9] discusses a rule-based policy management system that may support a policy-aware infrastructure in an open environment such as the World Wide Web. It describes the use of rich representational framework offered by the Semantic Web [6, 7, 8] technologies where it may allow both people and machines to "play by the rules" for social and online interactions. The paper primarily focuses on a rule-based access mechanism based on the Semantic Web technologies such as Notation 3 (N3) [91] and CWM [97].

2.4 Communication and Architecture

A pipes and filters [98, 99, 17] provides a structure for systems that process a stream of data. It consists of a set of components, or filters, connected by pipes so that each component reads stream of data on its inputs and produces streams of data on its output. A pipeline [17, 18] is a specialization of pipes and filters that restricts the topologies to a liner sequence of processing components. A Unix shell script with multiple programs is the best known example for a pipeline where each program may be connected by pipes.

A compiler is often viewed as a pipeline where a lexical analyzer, a parser, a semantic analyzer, and a code generator are aligned by processing data streams.

Syslog [100] is a proposed Internet Engineering Task Force (IETF) standard protocol that allows log messages to be forwarded a syslog server. It may construct a pipeline in sending and receiving log data streams in a Unix-based environment where a log writer and reader may be aligned with other processing components.

Common Object Request Broker Architecture (CORBA) [101, 102] is a standard middleware technology by Object Management Group (OMG) [60] that allows distributed objects to communicate in a heterogeneous computing environment. Interoperability relies on an Interface Definition Language (IDL) specification defining data types and object interfaces. The distributed objects implemented in diverse programming languages such as C, C++, and Java, may integrate by language bindings where the IDL specifications are translated to target programming languages and system environments, and interoperate by Remote Procedure Calls (RPC).

W3C [10] describes "a Web service is a software system designed to support interoperable machine-to-machine interaction over a network [11]." By facilitating XML-based Object Access Protocol (SOAP), it supports RPC and messaging styles of integration among software components [16, 103, 104]. Similar to CORBA IDL, it may utilize Web Services Description Language (WSDL) that specifies the machine-readable description of interfaces for components such as data types and operations.

Representational State Transfer (REST) is initially introduced as an architectural style for building large-scale distributed hypermedia systems [13, 14, 15]. It is characterized with four principles: URI-based resource identification, uniform interface, self-

descriptive messages, and stateful interactions though hyperlinks [16]. "REST uses standard interfaces to promote stateless interactions by transferring presentations of resources, rather than operating directly on those resources [104]." While the traditional Web services, known as "Big Web Services", are often criticized for its complexity and proliferations of specialized interfaces, a REST-based approach, or "RESTful Web Services", is gaining an increased attention due to its uniform interface for reuse and the simplicity for publishing and consuming services [12, 16].

An agent is "an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives" [105]. Interaction is the most important single characteristic of complex software [106] where the complexity is managed by decomposition, abstraction, and organization [107]. An agent has properties such as autonomy making a decision, reactivity to situations, pro-activeness taking the initiative, and social ability for interactions with an agent communication language [108, 106]. Jennings et al [107, 106] describes that an agent-based approach is a favorable software engineering to develop a complex distributed system. It is a "natural way to modularize a complex system in terms of multiple autonomous components that can act and interact in flexible ways in order to achieve their set of objectives" [107].

Foundation for Intelligent Physical Agents (FIPA) [109], the standards organization for agents and multi-agent systems, proposes Agent Communication Language (ACL) [110] for agent communications. The ACL specification includes the interaction protocols, communicative acts, and content languages. Two agents communicating each other share a common ontology for understanding terminologies and concepts. FIPA

Semantic Language (SL) [110] is a content language that can formulate the message contents in ACL. Schiemann et al. [111] describes that separating the message content semantics in SL is difficult and is not interoperable with Semantic Web. The paper proposes OWL/DL as a content language for agents so that it may reuse existing ontologies while reducing integration problems. Pasha et al. [112] presents a translation approach to integrate FIPA SL and OWL for Semantic Web. Nguyen et al. [113, 114] presents a gateway-based approach to integrate Web Services to FIPA agent services where Web Services runs as agent services. Foster et al. 2004 [115] discusses the mutually supportive nature of both Grids and agents. Grid systems provide an infrastructure for federated, reliable, and secure resource sharing across domains while agent-based systems focus on autonomous problem solving in dynamic environments. Corkill [33] reviews two different approaches of collaborating-software paradigms, blackboard and multi-agent s systems. Blackboard systems support collaboration by indirect communications through a centralized medium, or blackboard, while multi-agent systems pursuit an agent-centric orientation where its goal is to achieve effective collaboration with a group of independent, distributed software entities. The paper presents a flexible architecture to construct a blackboard system in an agent-based environment.

2.5 Transfer Syntax

Abstract Syntax Notation One (ASN.1) [116, 117] is a data specification language jointly standardized by International Standard Organization (ISO) [118] and International

Telecommunication Union (ITU) [119]. It describes data structures in a machine-independent format and allows transmissions among distributed components using various encoding rules such as Basic Encoding Rules (BER), Distinguished Encoding Rules (DER), Packed Encoding Rules (PER), and XML Encoding Rules (XER). BER supports flexible multiple encodings using 8-bit octets. DER allows a unique encoding that is needed for a hash in security. PER is a compact format for BER. XER facilitates an XML-based encoding scheme.

XML [55] is a general purpose markup language that describes structured data and serializations where XML Schema [56] may specify the valid data types and structures. WSDL facilitates XML Schema to specify the data structures and types. OWL XML Presentation Syntax (OWL/XML) [120] provides a XML Schema defining a XML presentation for an OWL-based ontology. XML CORBA Environment-Specific Inter-ORB Protocol (XIOP) [121] facilitates an XML-based representation to communicate an Object Requester Broker (ORB).

Javascript Object Notation (JSON) [122, 123] and YAML Ain't Markup Language (YAML) [124] facilitate the syntax that does not rely on markups. JSON is a text-based lightweight data interchange format that can represent language-independent and human-readable data structures. With its simple syntax, it is often used to as an alternative to XML for transmitting structured data over network. YAML is also a human-readable and language-independent data interchange format. Ben-Kiki et al. [125] describes that JSON's primary design goal is simplicity and university at the cost of reduced human readability. YAML can be viewed as a superset of JSON that offers improved human

readability and a more complete information model supporting any arbitrary data structures.

The FIPA ACL [109, 110] specification provides multiple message representations that can be serialized in Bit-Efficient, String, and XML. The serialized ACL message may be delivered using transport protocols such as CORBA Internet Inter-ORB Protocol (IIOP), Wireless Access Protocol (WAP), or HTTP.

2.6 Negotiation

Zulkernine et al.[126] proposes a Negotiation Broker (NB) middleware that supports automated negotiations of Service Level Agreement (SLA) for Web Services. Negotiating parties express the high-level policy specifications containing different parameters such as goals, contexts, constraints, issues, and values. The policy specification is represented in WS-Policy [127], and the protocol uses the FIPA Contract Net Interaction Protocol [109]. The trusted broker maps the policy specifications to negotiation strategies and parameters, and it performs the negotiations. Hung et al [128] proposes an independent declarative XML language, WS-Negotiation, for Web Services providers and requesters. The paper describes WS-policy is for a broad range of service requirement, preferences, and capabilities that lacks a negotiation solution for Web Services. It discusses negotiations in terms of Message, Protocol, and Decision-Making as well as a SLA template model supporting business negotiations for different domains.

Su et al. [129] describes an Internet-based negotiation server conducting bargaining-type negotiations for e-commerce. A negotiation server is registered with requirements,

constraints, and rules for negotiation policies and strategies that are represented with Object-Oriented Content Specification Language (OOCSL) and Event-Trigger-Rule (ETR) specification [130]. Gimpel et al. [131] discusses Policy-driven Automated Negotiation Decision-making Approach (PANDA). A negotiation policy is represented as the combination of XML-based rules and utility functions, and it is executed by decision-maker components.

Khedr et al. [132] and Al Ridhawi et al. [133] present an ontology-based negotiation protocol for Context-Level Agreement (CLA). The protocol is defined upon a context-aware system that is based on OWL to represent various negotiation contexts including messages, profiles, and rules.

Chapter 3 - Whiteboard Component

A whiteboard is a medium that provides full information sharing among teams. It supports team coordination by allowing them to achieve awareness of dynamic situations. In human organizations, it may contain a physical collection of messages, notes, and other artifacts to be shared. In the framework, it is a service-based information repository that facilitates Semantic Web technologies. Each team may have a whiteboard. As such, teams cope with distributed context information for coordination.

3.1 Ontology-based Data Representation

A whiteboard facilitates ontologies to represent context information. In contrast to semi-structured or free-text logs, a whiteboard contains fully-structured event logs, policies, and registry information. Its content structures and semantics are defined with a reference to ontologies for a problem domain. Ontology is a platform-independent expressive model that represents the shared concepts for a domain with classes, properties, and constraints. It may represent context that is any information, used to characterize the situation of an entity [1, 50]. As such, the whiteboard contents describe contextual information based on objects and their relationships.

A whiteboard adopts two-stage definition for ontologies. The first stage defines a generic ontology, *Ont-Team*, for core concepts relevant to a team coordination environment. The second stage extends the generic ontology to a specialized problem

domain. This approach allows core concepts to be reused in different domains while it may increase potential interoperability among domain-specific ontologies. Figure 3-1 highlights a sample concept hierarchy that is defined across two ontologies.

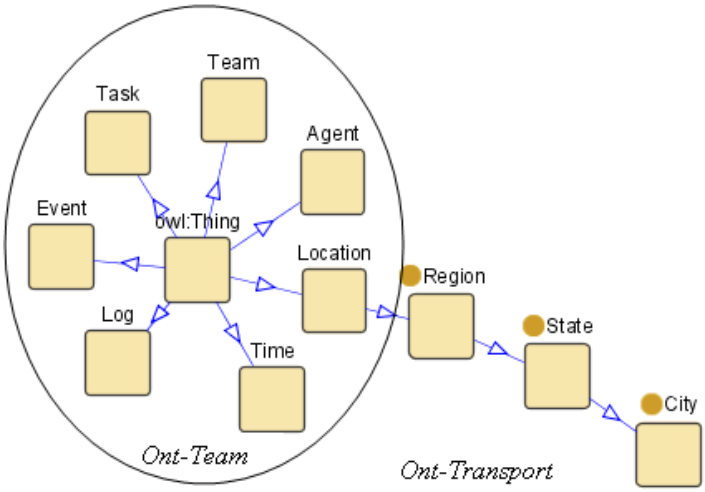

Figure 3-1 Two-Stage Definition of Ontologies

Figure 3-2 shows the relationships among core concepts in the *Ont-Team* ontology. *Team* has a relationship with *Agent*, *Location*, and *Task*. It models an abstract context where a team of agents works on a task in a certain location. An event log is represented as a combination of *Event* and *Log*, jointly having the relationships to *Agent*, *Location*, *Time*, and *owl:Thing*. With *owl:Thing*, an event log may record any activity for an agent such as a task because all concepts inherit it by default. A whiteboard may collect all contexts represented with these event logs.

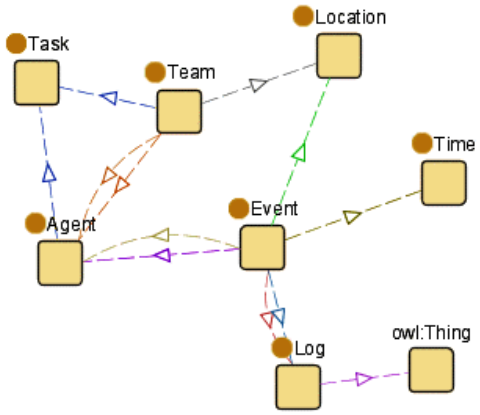

Figure 3-2 Relationships in Ont-Team Ontology

The *Ont-Transport* is an ontology that extends the *Ont-Team*, describing a domain-specific environment for the *Transport* example in Chapter 6. Figure 3-3 shows a sample display of the whiteboard content data that are the instances of the *Ont-Transport* ontology.

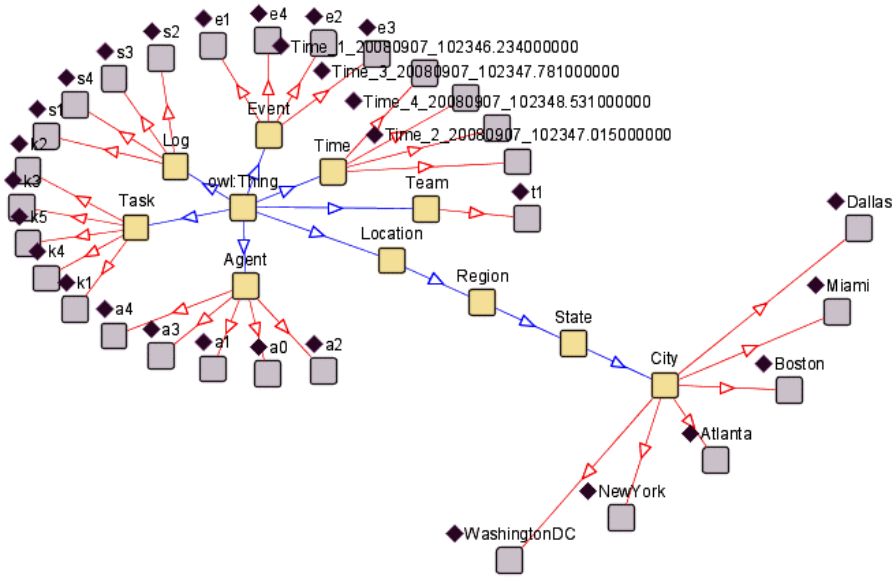

Figure 3-3 Whiteboard Contents - Ont-Transport Ontology Instances

3.2 Policies and Queries

A whiteboard may contain policies and queries facilitating Semantic Web Technologies. Policies are the rules governing an organization and team situations. It utilizes SWRL rules and SQWRL queries that can be applied to OWL-based contexts. For example, the following SWRL rules describe that *a city, x, has a transportation to a city, y, if there is either a flight or a bus from x to y.*

- City(?x) ^ City(?y) ^ hasFlightTo(?x, ?y) -> hasTransTo(?x, ?y)
- City(?x) ^ City(?y) ^ hasBusTo(?x, ?y) -> hasTransTo(?x, ?y)

With a SQWRL query, *all cities, having any transportation from x to y*, may be retrieved:

- City(?x) ^ City(?y) ^ hasTransTo(?x, ?y) -> sqwrl:selectDistinct(?x, ?y)

Any indirect transportation may be further retrieved with an additional rule.

- City(?x) ^ City(?y) ^ City(?z) ^ hasTransTo (?x, ?y) ^ hasTransTo(?y, ?z) -> hasTransTo(?x, ?z)

Team situations may be indentified by reasoning. Reasoning allows low-level contexts to be inferred to high-level contexts or situations. For instance, an urgent team situation may be inferred by rules using known context information. Policies may define that *a team is in an urgent situation if it performs a task that has a priority greater than a certain value*.

- Team(?t) ^ Task(?k) ^ hasPriority(?k, ?p) ^ (?p > 7) -> isUrgent(?t, true)

Policies and queries are managed in a separate whiteboard and accessed by references. This allows alternative policies to be declared and adapted to a different environment. It may further cope with policies or rules in a distributed organization [134] where several set of rules may exist based on an organizational structure.

The framework additionally supports a special coordination query, *coquery*, that is based on distributed contexts and rules from multiple whiteboards. The details are further discussed in Chapter 6.

3.3 Whiteboard Types

The framework separates the types of managed information to enhance flexibility, reusability, and interoperability. By separating an ontology-based schema from other contents, the same schema may be referenced. As such, it is reusable, potentially increasing a semantic interoperability among distributed contexts. By separating policies, alternative policies may be flexibly adapted to a specific environment. Essentially, this separation provides a foundation for whiteboard-based coordination where distributed contexts are shared with a policy-aware protocol.

The framework supports several different types of whiteboards that are distinguished by the type of managed information. ONTWB (O) contains ontology-based schema that defines the content structure and semantics, such as classes, properties, relationships, and constraints. Other types of whiteboards are instantiated, and their contents are validated with the ONTWB. LOGWB (W) is to store the event logs, produced by agents in a team

working environment. It is an information store for maintaining situational information or contexts. POLWB (P) is a collection of policies that are rules governing an organization and situations. It may contain a set of predefined queries and rules to reason about a high-level context or team situation with event logs in LOGWB. COWB (C) is a temporary workspace for a coordination query, *coquery*, that binds distributed contexts and rules from multiple whiteboards. REGWB (R) maintains resource registry information about teams or agents, similar to a directory service.

3.4 Whiteboard Operations

A whiteboard allows a generic set of operations that manage objects, properties, and relationships. A whiteboard component supports these operations with APIs allowing other application or components to integrate. The operations include object-level operations including *create* and *delete* as well as property-level operations including *set* and *get*. A high-level operation for building blocks may utilize this generic set of operations in a diverse way. A few generic operations in pseudo syntaxes are as follows:

- *w <- new WB(m)*

  Instantiate a whiteboard, *w*, with an ontology-based schema, *m*.

- *o <- w.add(c [,n])*

  Create or update an object, *o*, of a class, *c*, with an optional name, *n*.

- *o.delete() or w.delete(n)*

  Delete an object, *o,* by reference, or by name, *n*.

- *o.set(p, v)*

  Set a property, *p* with a value or an object, *v*.

- *v <- o.get(p)*

  Get a value or an object property, *p*.

The whiteboard also supports rules and queries that may be expressed with SWRL and SQWRL. It provides a set of APIs including *query* and *coquery* that may perform a query with rule-based reasoning. The *coquery* is a special query that can be triggered with multiple whiteboard references. The following lists several APIs in pseudo syntaxes:

- *w.addRule(r [,n])*

  Add a rule, *r*, with an optional name, *n*.

- *w.deleteRule*(n)

  Delete a rule with a name, *n*.

- *w.enableRule(n)*

  Enable a rule with a name, *n*.

- *w.disableRule(n)*

  Disable a rule with a name, *n*.

- *s <- w.query(q [,R])*

  Perform a query, *q*, optionally with a set of rules, *R*.

- *s <- w.coquery(q, W, P)*

  Perform a query, *q*, with a set of LOGWBs, *W*, and a set of POLWBs, *P*.

3.5 Implementation Layer

A whiteboard is a service-based component built with the Semantic Web technologies. It provides a simple set of APIs that is layered on top of Protégé-based OWL, SWRL, and SQWRL API libraries [96] with a bridged JESS rule engine [135]. It supports a generic set of operations to manage the contents, represented by objects, properties, and relationship as well as rules and queries. The whiteboard APIs may be exposed as Web Services APIs so that any external components or applications may integrate a whiteboard for interactions.

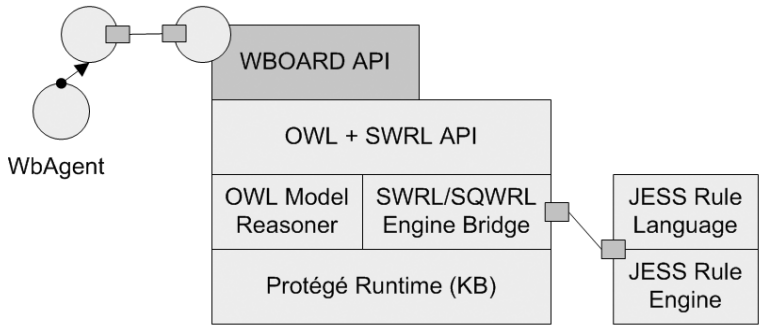

Figure 3-4 Whiteboard Implementation Layer

Chapter 4 - Communication and Architecture

This chapter discusses the communicational and architectural perspective of the framework. The framework utilizes a hybrid approach to supports flexible yet inoperable communications for a team working environment. It provides several communication styles that are required for diverse interactions and collective system behaviors by agents. The discussion includes an overall architecture to implement an agent with a whiteboard as well as logical layers of system components.

4.1 Computing Environment

The framework facilitates an agent-based approach. Its environment is modeled with distributed multiple agents as shown in Figure 4-1. An agent is an autonomous computing entity that is situated in an environment with a specific role. It is reactive to the changes in the environment and capable of making a decision. Agents may interact with other agents to cope with different situations while accomplishing a goal. A simple interaction may be a message delivery or a service request. A complex interaction may require a sophisticated collective behavior, such as distributed context sharing and negotiation.

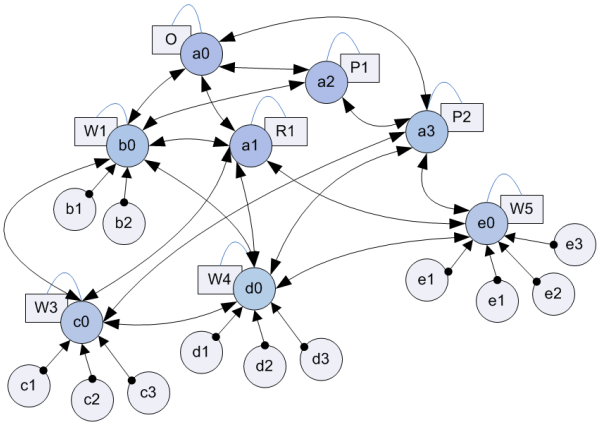

Figure 4-1 Agents

4.2 Problem Domain: Logical Layout

An agent acts individually or establishes a relationship with other agents on behalf of organizational context. Figure 4-2 shows the logical layout of agents that shows the relationships in teams and interactions to other service entities. An agent may independently work on a task, such as providing a registry service, or it may be a peer working in a team either with a leader or a member role. Teams may interact for collaboration.

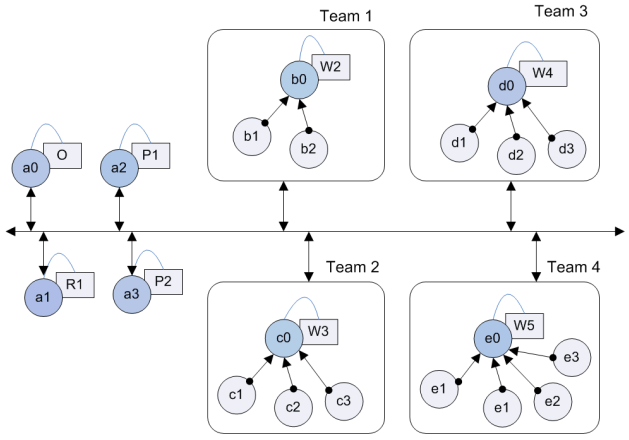

Figure 4-2 Logical Layout of Agents and Teams

4.3 Agent Types

Figure 4-3 shows several different types of agents. An agent type and its associated whiteboard determine an agent's role and its expected behaviors. The whiteboard types are discussed in Chapter 3.

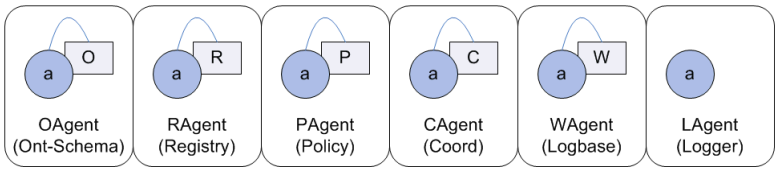

Figure 4-3 Agent Types

OAgent(OA) is a service agent with ONTWB (O) for ontology-based schemas. It supports other whiteboards to be instantiated and populated with valid content structures. WAgent(WA) and LAgent(LA) are logbase and logger agents that play team leader and member roles, respectively. A LAgent collects event logs in an environment and sends them as messages to the WAgent. The WAgent translates them and stores the log contexts in its LOGWB (W). PAgent(PA) provides a policy service with POLWB (P) that contains rules governing team situations in an organization. RAgent(RA) utilizes REGWB (R) managing registry information about teams and agents. CAgent(CA) is a special agent role as a coordinator that is required when teams needs to negotiate. It facilitates a temporary workspace, COWB (C), and triggers a coquery using multiple whiteboards. Any participant or a trusted 3rd party may play this role, depending upon the nature of an organization. In the framework, a team leader initiating negotiation is assumed to play this role.

4.4 Collective System Behavior

The framework provides the advanced system functions by interactions and cooperation among agents. While a single agent can play a specific role based on its type, a group of agents can work together to cope with more complex behaviors. The flexibility of the framework, designed with an agent as a networked entity, an ontology-based whiteboard, fully-structured log messages, and multiple communication options, can provide a foundation for modeling different interactions and system behaviors. The following highlights several collective system behaviors in the framework:

- Event Logging System: {OA, WA, LA, *log*}
- Search System: {OA, WA, *query*}
- Rule-based System: {OA, WA, PA, *query*}
- Registry System: {OA, RA, *query*}
- Distributed Context Sharing System: {OA, m*WA, n*PA, *coquery*}

4.5 Communication Styles

The framework utilizes a hybrid approach to support flexible communication for different interactions and collective system behaviors. It facilitates Web Services [10, 11, 12], Representational State Transfer (REST) [13, 14, 15, 16], and pipeline [18, 99, 17]. Figure 4-4 shows several communication options supported in the framework.

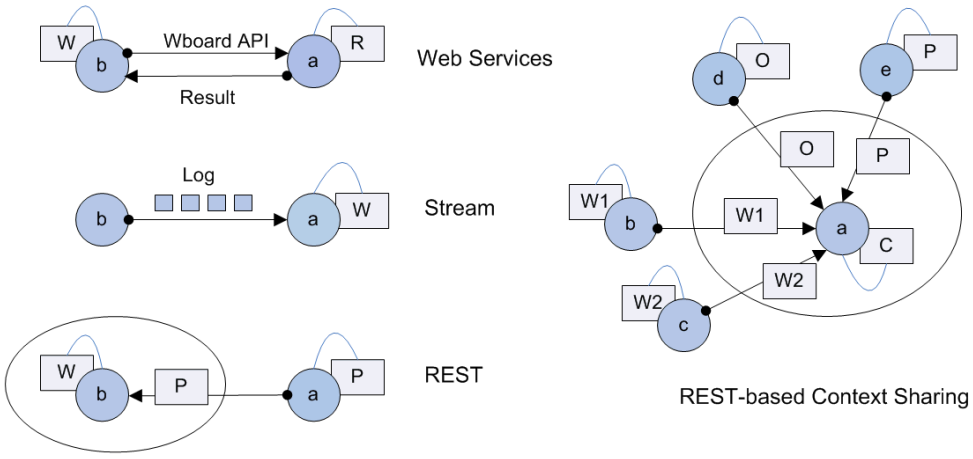

Figure 4-4 Communication Styles

4.5.1 Web Services

The SOAP-based Web Services provide the RPC-style APIs for whiteboard operations. In the framework, an agent may facilitate a whiteboard and its operations to manage various contents. An agent exposes these operations as Web Services APIs so that other agents or components may interact and perform operations on a whiteboard. Essentially, the framework supports a service-based communication infrastructure among agents and components.

4.5.2 REST

The framework facilitates a REST-style of communication to support the sharing of distributed context and rules, allowing awareness of situations and policies. In contract to Web Services, REST "uses standard interfaces to promote stateless interactions by transferring presentations of resources, rather than operating directly on those resources

[104]." A whiteboard is an information resource that can be represented in a self-descriptive message using the OWL/XML presentation syntax [120]. The framework devises this REST-style of communication to share contexts and rules in whiteboards.

4.5.3 Pipeline

The framework may handle log streams through a pipeline by arranging multiple components into an I/O-based chain. CELL is a line-based language that can represent a fully-structure event log. A CELL translator interprets CELL streams to whiteboard operations. A CELL line may be enveloped within a Syslog [100] message that is a standard protocol for distributed logging. As such, by pipelining a Syslog filter and a CELL translator, a CELL-based event log messages may by delivered using Syslog. This potentially enables the framework to communicate though an existing logging infrastructure.

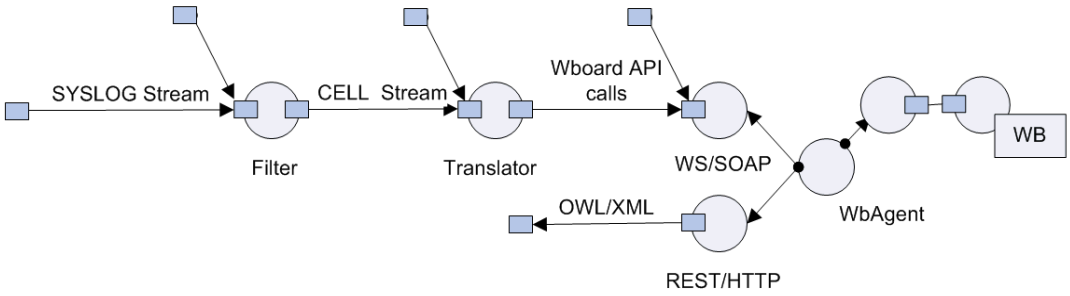

Figure 4-5 Pipeline - Log Streams

4.6 Implementation Architecture

Figure 4-6 shows an overall implementation architecture for an agent in the framework. Several components are defined under two groups, WBOARD AGENT and WBOARD LAYER, that are interconnected with an adaptor. The WBOARD AGENT primarily handles communications and services. The WBOARD LAYER provides whiteboard operations as described in Chapter 3.

The WbAgent is the main application that controls all other components. It is designed with embedded lightweight SOAP and HTTP server to support Web Services and REST-style interactions. The Filter may extract CELL messages from Syslog steams, and the translator interprets them to whiteboard operations to store log contexts.

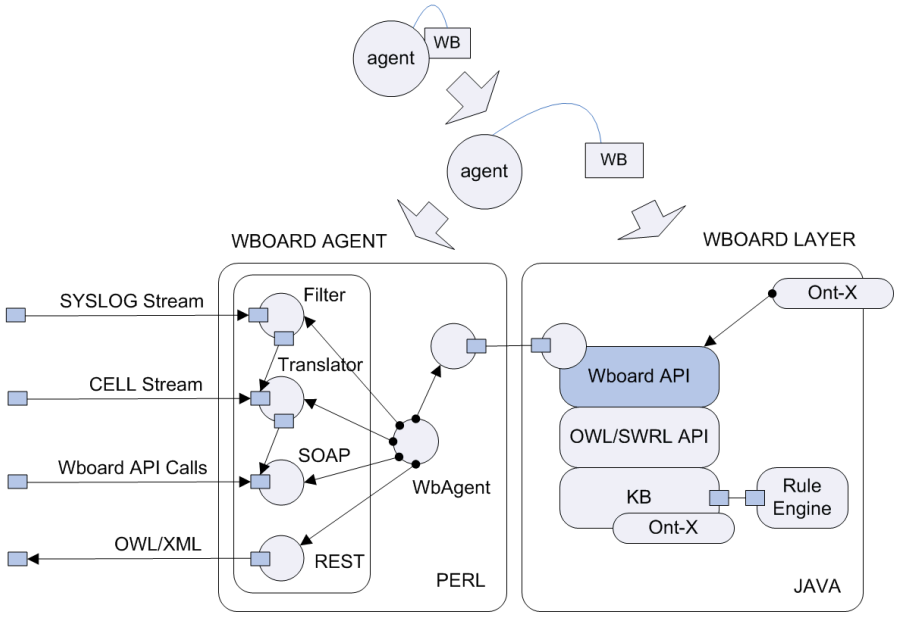

Figure 4-6 Agent Components

4.7 Logical Layers of System Components

All system components in the framework may be logically viewed as layers as shown in Figure 4-7. A component in an upper layer utilizes the lower layer functions or services. The Wboard APIs facilitate the underlying Semantic Web technologies, and it only allows internal interactions with upper layers. The Web Services and REST supports external interactions. The CELL Translator and the Syslog Filter further utilize their lower layers to provide external interfaces.

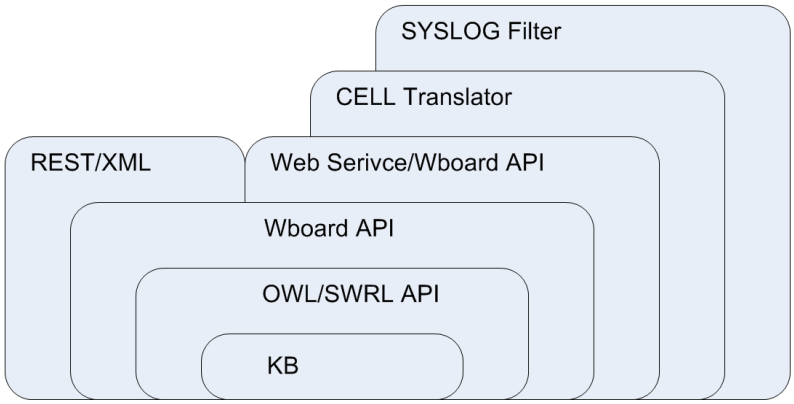

Figure 4-7 System Component Layers

Chapter 5 - Context-aware Event Log Language

Context-aware Event Log Language (CELL) is a simple line-based scripting language that may specify a fully-structured even log based on ontologies. The framework facilitates CELL as transfer syntax to deliver event logs as data streams. It is designed with simple concise syntax to support both human and machine readability. A distributed application or an agent may send CELL-based messages, and a whiteboard agent interprets to whiteboard operations using an embedded translator.

5.1 5W1H-based Event Log Structure

An event log is the description about contexts or situations in an environment. In contrast to the traditional logs that are semi-structured or free text, the framework facilitates a fully-structured representation to preserve precise event semantics. An application or an agent generates an event log message and sends it to a whiteboard. A whiteboard agent stores it by translating to whiteboard operations. A shared ontology is referenced by both of an event log message and a whiteboard. As such, the semantics of the message context are preserved throughout a log transfer.

In the framework, an event log is conceptually described as a vector of five features that denote a 5W1H:

- <*Who*, *What*, *Whom*, *Where*, *When*, *How*>.

The *Who* indicates a sender agent that generates a log message. The *What* is a descriptive log context. It is fully-structured, conforming to an ontology in ONTWB. The *Whom* indicates a receiver that is a whiteboard agent with LOGWB. The *Where* and *When* describe the location and time. The *How* may optionally describe other extra information to be logged.

Figure 5-1 visualizes a 5W1H-based event log in a data structure and ontology. In a traditional programming language such as C, it may be represented with linked-lists or a similar data structure. With an ontology-based model, it is represented as relationships among objects and properties.

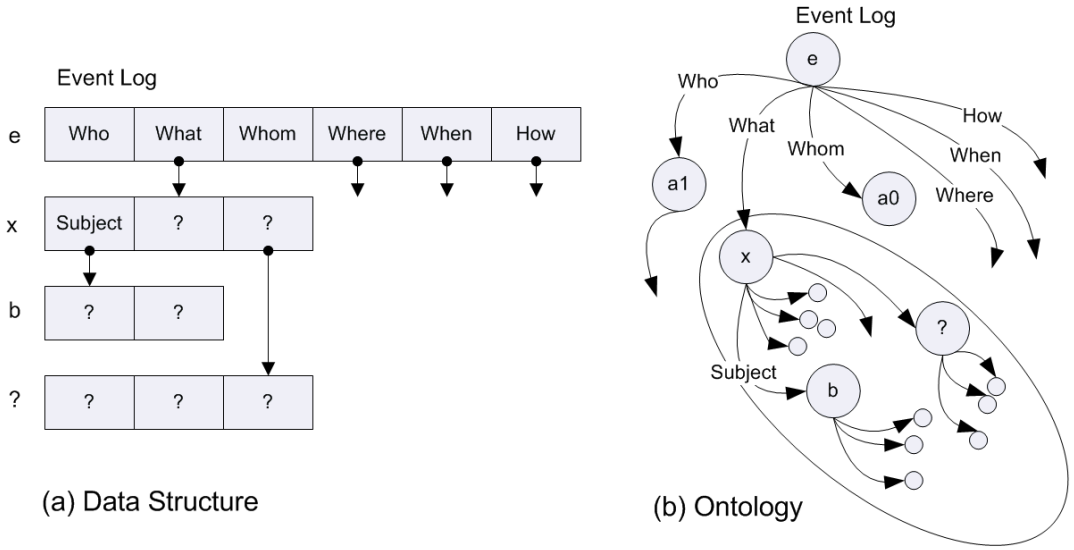

Figure 5-1 Data Structure and Ontology

The following line shows a 5W1H-based event log, describing that "*an agent, a1 in Boston, sends an event log message that envelops a context, x, to other agent, a0, at 2008-07-16T13:24:56"*.

- <a1, {x | x is a context}, a0, Boston, 2008-07-16T13:24:56, …>

The context, *x*, may be a detailed description such that "*Boston has a flight to New York*". A corresponding context may be represented in CELL:

- {Boston | hasFlightTo={NewYork}}

5.2 Comparison

This section illustrates a few examples to describe the differences of CELL and other representations such as traditional logs and XML.

5.2.1 Traditional Logs and CELL

For a traditional log with a free text or semi-structured format, a data extraction step is required in order to enable a machine to interpret the log context. For instance, a following line shows a log message in the RFC 3164 specification for Syslog [100]:

- Aug 16 13:24:56 [a1:a0]: Boston has a flight to New York

The log line is semi-structured such that a descriptive context, "*Boston has a flight to New York*", is in a free text while other fields such as a timestamp, "*Aug 16 13:24:56*", and a log source, "*a1:a0*", may be structured by order in the line. To enable a machine to

understand the message context, its terms and structures need to be extracted in a machine-readable format such as a document vector [136, 137]:

- <Boston, flight, New York>

However, the extracted representation may not precisely reflect a relationship among terms. It is not clear if the *flight* is a relationship between *Boston* and *New York* in a flat vector. Similarly, a more descriptive context, "*the Boston city in MA has a flight to New York*", may be represented as follows:

- <Boston, city, MA, flight, New York>

It does not describe a relationship between two terms, *Boston* and *city*, nor *Boston* and *MA*. It is not clear if the term, *New York*, is a city or a state.

A CELL-based line may represent a fully-structured semantic log context. For instance, the previous two lines may be expressed in CELL:

- {Boston | hasFlightTo={NewYork}}
- {Boston:City | state=MA, hasFlightTo={NewYork:City}}

The first line describes a relationship, *hasFlightTo*, between two objects, *Boston* and *NewYork*. The second line represents that *Boston* belongs to a state, *MA*, and *NewYork* is of a class, *City*. Its valid structure relies on an ontology-based schema.

5.2.2 XML and CELL

A 5W1H-based event log message may facilitate OWL/XML Presentation Syntax [120, 96] for log transfer syntax. XML [55] is a general purpose markup language that may describe structured data and transfer syntax. It uses a document-oriented tree structure with machine-friendly syntax. It is logically represented as a hierarchical tree where each node is a block enclosed by a pair of tags. The OWL/XML Presentation Syntax supports OWL-based ontologies to be serialized using XML. Figure 5-2 shows an event log in the OWL/XML Presentation Syntax.

Log messages are generally delivered as text streams or lines using a standard logging protocol such as Syslog. A pipeline may be constructed with other components in order to dynamically process log streams on the fly. For example, only log messages from a particular source or event may be selectively filtered for special handling such as the analysis of a urgent condition.

An XML-based approach does not fit well for representing message streams or lines. Its document-oriented tree structure produces a long and nested message representation with markups. Assembling a pipeline with I/O-based text streams is complex, compared to a line-based approach. Furthermore, this XML-based approach has a limited human readability.

CELL facilitates a simple line-based representation. Compared to XML, CELL is far more simple and concise, and it produces both machine-readable and human-friendly representation as shown in Figure 5-3. CELL-based messages may be transferred as

streams and easily assembled for a pipeline. A CELL line may be enveloped within a Syslog message as it is also line-based.

```
<Event rdf:ID="e1">
  <who rdf:resource="#a1"/>
  <when>
    <Time rdf:ID="Time_1_20080826_105833.375000000">
      <timestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >2008-07-16T13:24:56</timestamp>
    </Time>
  </when>
  <whom rdf:resource="#a0"/>
  <what>
    <Log rdf:ID="s1">
      <subject>
        <City rdf:ID="Boston">
          <flightTo>
            <City rdf:ID="NewYork">
              <region rdf:datatype
                    ="http://www.w3.org/2001/XMLSchema#string"
              >East Coast</region>
              <state rdf:datatype
                    ="http://www.w3.org/2001/XMLSchema#string"
              >NY</state>
              <city rdf:datatype
                   ="http://www.w3.org/2001/XMLSchema#string"
              >New York</city>
            </City>
          </flightTo>
          <region rdf:datatype
                  ="http://www.w3.org/2001/XMLSchema#string"
          >East Coast</region>
          <state rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >MA</state>
          <city rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Boston</city>
        </City>
      </subject>
    </Log>
  </what>
  <where rdf:resource="#Boston"/>
</Event>
```

Figure 5-2 Event Log in OWL/XML

```
{ Boston:City  | city=Boston, state=MA, region="East Coast" }
{ NewYork:City | city="New York", state=NY, region="East Coast" }

{ e1:Event |  \
    who={a1}, \
    what={s1:Log | subject={Boston | flightTo={NewYork}}}, \
    whom={a0}, \
    where={Boston}, \
    when={:Time | timestamp="2008-07-16T13:24:56"} \
}
```

Figure 5-3 Event Log in CELL

5.3 CELL Language Specification

The primary objective of CELL is to represent a fully-structured event log and to utilize it as log transfer syntax in the framework. Its design considerations are as follows:

- Simple and concise syntax
- Line/Stream-based
- Machine-readable and human-friendly
- Deliverable by Syslog
- Ontology-based
- Support for scripting and testing

5.3.1 Object Specification

An object is a symbolic representation of a thing in the real world. It may be associated with a set of properties that characterizes an object. An object specification in CELL is simple and concise. The following example line represents "*an object, o1, of a*

class, c1, such that its property, p1, has a value, v1, and it has a relationship, p2, to an object, o2".

- {o1:c1 | p1=v1, p2={o2}}

The following EBNF describes the language syntax for an object specification:

<object> ::= '{' (<named-object> | <unnamed-object>) '}'

<named-object> ::= <object-name> [':' <class-name>] ['|' <property-list>]

<unnamed-object> ::= ':' <class-name> ['|' <property-list>]

An object is enclosed with a pair of curly braces that collects its properties. Each object may be either named or unnamed. A named object is additionally preceded by an identifier, followed by a vertical bar. It may be optionally specified with a class name to denote a type of an object.

5.3.2 Property Specification

A property describes an object's characteristics and consists of a key-value pair. The EBNF syntax for properties is shown as follows:

<property-list> ::= <property> { ',' <property> }

<property> ::= <prop-key> '=' <prop-val>

<prop-key> ::= [<object-name> '.'] <identifier>

<prop-val> ::= <literal> | '[' <prop-val-llist> ']' | <object> | <prop-key>

<prop-val-list> ::= <prop-val> { ',' <prop-val> }

The actual value may be a scalar, a list, or an object. The <*property-list*> is a collection of properties that may be used for an object specification. Several examples of property expressions are as follows:

- p1 = v1
- p1= “a value”
- p1= [v1, v2, {o2}}]
- p1={o1}

The *p1* is a named key. Because CELL is dynamically typed, a value, *v1*, can be of any primitive type including integer, real, date, and string, and quoted literal. The multiple values are represented with a list where each element can be recursively specified using any of above values. A property may have an object, *{o1}*, describing a relationship, *p1*, to other object, *{o1}*.

5.3.3 Named Object

A named object has a unique object identifier, allowing other objects to reference it. A specification with an optional class name may create a new object instance. If a named object already exists, it only refers to the existing object, but its type is validated. An object without a class is for a reference only. Several examples of named objects are as follows:

- {o1 | p1=v1}
- {o2:c1 | p1=v1, p2=[v2,v3], p3={o1}}

5.3.4 Unnamed Object

An unnamed object may be specified within other object expression. It is required to specify a class name such that a new instance is always created. It may be only referenced indirectly though other object's property, or a relationship:

- {o3 | p1=v1, p3={:c1 | p1=v1}}

5.3.5 Nested Object

Both named and unnamed object specifications may be nested within other object expressions. It may be recursively defined to represent a complex object. For example, the following examples show the nested object specifications.

- {o4 | p1=v1, p3={o1:c1 | p1=v1}}
- {o5 | p1=v1, p3={o6 | p1=v1, p3={o7 | p1=v1}}}
- {o8 | p1=v1, p3={:c1 | p1=v1, p3={:c1 | p1=v1}}}

An object, *o5*, may be equivalently expressed with a list of named objects as follows:

- {o7 | p1=v1} {o6 | p1=v1, p3={o7}} {o5 | p1=v1, p3={o6}}

5.3.6 Rules and Queries

CELL supports SWRL rules and SQWRL queries that may be directed embedded in CELL. It is an enhanced capability to support a scripting and testing environment for a

whiteboard. The following EBNF shows the language syntax for embedded SWRL rules and SQWRL queries:

<rule> ::= '>{' (<named-rule> | <unnamed-rule>) '}'

<named-rule> ::= <rule-name> '|' <swrl-rule>

<unnamed-rule> ::= <swrl-rule>

<rule-cmd> := (enable-rule | disable-rule | delete-rule) <rule-name>

<query> ::= '>{' (<named-query> | <unnamed-query>) '}'

<named-query> ::= <query-name> ['|' <sqwrl-query>]

<unnamed-query> ::= <sqwrl-query>

<query-cmd> := (enable-query | disable-query | delete-query) <query-name>

Similar to an object specification, a rule may be either named or unnamed. A named rule has a unique identifier that may be explicitly referenced in a CELL script. An unnamed rule is only used for reasoning.

- %{ r1 | p(?x, ?y) ^ p(?y, ?z) -> p(?x, ?z) }
- %{ p(?x, ?y) ^ p(?y, ?z) -> -> p(?x, ?z) }

A query is similar to a rule except it actually returns results. A name query may be predefined and explicitly referenced later for execution. An unnamed query is only specified as a part of a query execution.

- %{ q1 | p(?x, ?y) ^ p(?y, ?z) -> sqwrl:select(?x, ?z) }
- %{ p(?x, ?y) ^ p(?y, ?z) -> sqwrl:select(?x, ?z) }

- %{ q1 }

5.3.7 Coquery

CELL provides a special query command, *coquery*, that utilizes distributed contexts and rules from multiple whiteboards. The following EBNF shows its syntax:

<co-query> ::= '>{' coquery <query> <logwb-uri-list> <polwb-uri-list> '}'

<logwb-uri-list> ::= <uri-list>

<polwb-uri-list> ::= <uri-list>

<uri-list> ::= <uri> { ',' <uri> }

In a sample usage synopsis below, it may take 3 sets of arguments. The *query* string may be either named or unnamed as described in the previous section. The second and third arguments are the whiteboard URLs of LOGWBs and POLWBs.

- >{ coquery %{query} [URL-W1, URL-W2, ...] [URL-P1, URL-P2, …] }

5.3.8 References: Whiteboard Service, Ontology, and Namespace

A CELL script is required to specify several references as a preface. These include the reference to whiteboard services, ontology-based schema, and namespaces. A CELL script is dynamically translated using a whiteboard service, and its contents are validated by ontology-based schema and namespaces. The following EBNF shows these reference syntax:

<preface> ::= <ws-ref> <ont-ref> <ns-ref>

<ws-ref> ::= [<line-tag>] '('<ws-alias>')' '@@' WB_SERVICEURI <uri>

<ont-ref> ::= [<line-tag>] ['('<ws-alias>')'] '@@' METAONTURI <uri>

<ns-ref> ::= [<line-tag>] ['('<ws-alias>')'] '@@' NSBASEURI <uri>

<ws-alias> ::= <identifier>

Sample reference specifications are as follows. Note that *w1* is an alias of a whiteboard service that may simplify scripting when multiple services are explicitly referenced in a single script.

- (w1) @@WB_SERVICEURI http://localhost/wb1
- @@WB_METAONTURI http://localhost/ontologies/transport/meta.owl
- @@WB_NSBASEURI http://wboard.team.org/ontologies/transport

5.3.9 Special Features: Line Tag, Whiteboard Alias, and Block

CELL provides experimental features to support the delivery of log messages using Syslog. These special features include line tags, whiteboard aliases, and block specifications. The following EBNF shows the corresponding language syntax:

<cell> ::= <preface> <line-spec> { <line-spec> }

<line-spec> ::= [<line-tag>] <line>

<line> ::= ['(' <ws-alias> ')'] <construct> | '(' <ws-alias> ')' <block>

<construct> ::= <directive> | <object> | <rule> | <query> | <co-query>

<block> ::= '<<' (__START__ | __END__)

<line-tag> ::= '~%' <literal>

With line tags, CELL streams may be filtered from several sources. Each CELL line may be optionally specified with a whiteboard alias so that it can easily reference a different whiteboard service. A block creates a region in a CELL script that may utilize the same whiteboard service.

5.3.10 Comments and Multi-line specification

CELL supports comments and multi-line specifications. Any portion of a line after '#' marker is treated as comments. A continuation marker, '\', allows a single line to be broken down to multiple lines. It helps a CELL script to be organized into a human-friendly format.

5.3.11 Utility Commands

Several utility commands are supported in the language: sync, flush, echo, and exit.

<directive> ::= '>{' <system-cmd> | <rule-cmd> | <query-cmd> '>'

<system-cmd> :: = <flush-cmd> | <sync-cmd> | <echo-cmd> | <exit-cmd>

<flush-cmd> ::= flush [true | false]

<sync-cmd> ::= sync [<path>]

<echo-cmd> ::= echo [<literal>]

<exit-cmd> ::= exit

The *sync* command forces the whiteboard contents in memory to be saved to a persistent storage. The *flush* command clears the whiteboard contents in memory, with an optional argument to wipe out the persistent storage as well. The *echo* command

prompts the message passed as an argument. The *exit* command forces the translator to terminate.

5.4 Translation and Whiteboard Integration

A whiteboard agent has an embedded translator that dynamically interprets a CELL script to whiteboard operations. The following example shows a CELL script and its translated whiteboard operations:

- {o1:c1 | p1=v1, p2={o2 | p3=v3, p4=v4 }}
    1. o2 <- wb.get(o2)
    2. o2.set(p3, v3)
    3. o2.set(p4, v4)
    4. o1 <- wb.add(c1)
    5. o1.set(p1, v1)
    6. o1.set(p2, o2)

The framework offers different ways to integrate a CELL script and a whiteboard. A CELL script may be passed as an argument when a whiteboard agent starts. As such, it supports a scripting environment, much similar to a command-line program. It may be also remotely processed using a whiteboard service in a distributed environment.

5.5 CELL EBNF

Figure 5-4 shows CELL language syntax in EBNF that are discussed in the previous sections.

```
<cell> ::= <preface> <line-spec> { <line-spec> }
<line-spec> ::= [ <line-tag> ] <line>
<line> ::= [ '(' <ws-alias> ')' ]  <construct> | '(' <ws-alias> ')' <block>
<construct> ::= <directive> | <object> | <rule> | <query> |  <co-query>
<block> ::= '<<' ( __START__ | __END__ )
<line-tag> ::= '~%' <literal>

<preface> ::= <ws-ref> <ont-ref> <ns-ref>
<ws-ref> ::= [ <line-tag> ] '('<ws-alias>')' '@@' WB_SERVICEURI <uri>
<ont-ref> ::= [ <line-tag> ] [ '('<ws-alias>')' ] '@@' METAONTURI <uri>
<ns-ref> ::= [ <line-tag> ]  [ '('<ws-alias>')' ] '@@' NSBASEURI <uri>
<ws-alias> ::= <identifier>

<object> ::= '{' ( <named-object> | <unnamed-object> ) '}'
<named-object> ::= <object-name> [ ':' <class-name> ] [ '|' <property-list> ]
<unnamed-object> ::= ':' <class-name> [ '|' <property-list> ]

<property-list> ::= <property> { ',' <property> }
<property> ::= <prop-key> '=' <prop-val>
<prop-val> ::= <literal> | '[' <prop-val-llist> ']' | <object> | <prop-key>
<prop-val-list> ::= <prop-val> { ',' <prop-val> }

<object-name> ::= <identifier>
<class-name> ::= <identifier>
<prop-key> ::= [ <object-name> '.' ] <identifier>

<directive> ::= '>{' <system-cmd> | <rule-cmd> | <query-cmd> '>'
<system-cmd> :: = <flush-cmd> | <sync-cmd> | <echo-cmd> | <exit-cmd>
<flush-cmd> ::= flush [ true | false ]
<sync-cmd> ::= sync [ <path> ]
<echo-cmd> ::= echo [ <literal> ]
<exit-cmd> ::= exit

<rule> ::= '>{' ( <named-rule> | <unnamed-rule> ) '}'
<named-rule> ::= <rule-name> '|' <swrl-rule>
<unnamed-rule> ::= <swrl-rule>
<rule-cmd> := ( enable-rule | disable-rule | delete-rule ) <rule-name>

<query> ::= '>{' ( <named-query> | <unnamed-query> ) '}'
<named-query> ::= <query-name> [ '|' <sqwrl-query> ]
<unnamed-query> ::= <sqwrl-query>
<query-cmd> := ( enable-query | disable-query | delete-query ) <query-name>


(Continue on next page ...)
```

```
(... Continued from the previous page)

<co-query> ::= '>{' coquery <query> <logwb-uri-list> <polwb-uri-list> '}'
<logwb-uri-list> ::= <uri-list>
<polwb-uri-list> ::= <uri-list>
<uri-list> ::= <uri> { ',' <uri> }

. . .
```

Figure 5-4 CELL EBNF

Chapter 6 - Situation/Policy-Aware Negotiation

In a dynamic team environment, collaboration among teams has to cope with unavailable or insufficient resources. A team may decide to collaborate with other teams on a task that can not be completed with the team's own resources. Collaboration with other teams allows the team to complete the task.

Collaboration requires an agreement among the teams. An agreement relies on a negotiation protocol to identify the teams that may participate in the collaboration and to decide on a partner. With multiple team candidates, the protocol has to assist with the final decision. Whether a team should collaborate with another team may be a measure that adapts to the organizational policies and the team situations.

This chapter presents a policy-based negotiation protocol, Situation/Policy-Aware Negotiation (SPAN), for finding a collaboration partner. It may adaptively cope with organizational policies and assist with a decision. Policies are the rules governing team situations in an organization where situations are represented with contextual information. The protocol relies ontology-based whiteboards as a computational foundation, and it facilitates Semantic Web technologies such as Web Ontology Language (OWL) [85, 87] and Semantic Web Rule Language (SWRL) [93, 94], and Semantic Query-enhanced Web Rule Language (SQWRL) [95]. The negotiation protocol assists with a decision by utilizing the combination of rule-based queries and functions to compute a measure. The rest of the chapter introduces an example scenario

describing a team situation demanding a collaborative work, and describes the logical steps of the protocol.

6.1. Simple Coordination Example

This section introduces a simple coordination example by describing a team working environment and a scenario that requires collaboration.

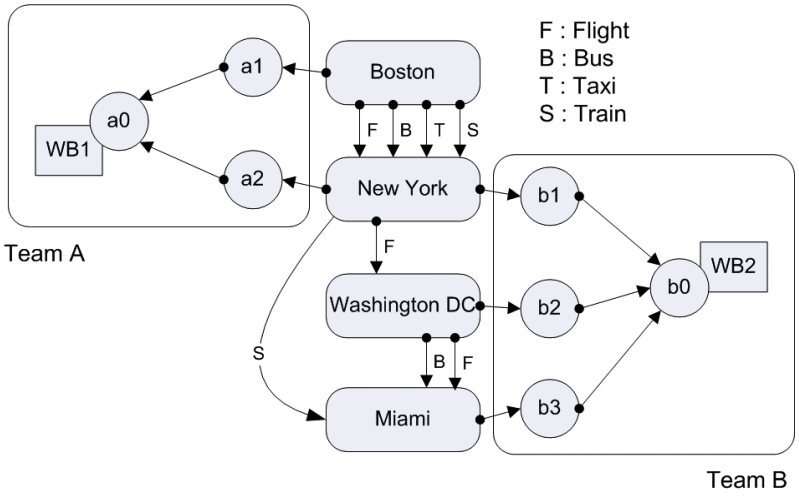

Figure 6-1 Transport Example

6.1.1 Transport - Team Working Environment

Figure 6-1 shows an example environment where teams track the traffic status among different cities. Each team consists of three or more agents: a leader and member agents. A leader has a whiteboard that collects event logs transmitted from member agents. A member agent reports to its leader on traffic status in a city. Therefore, a team only

knows about the status in a limited number of cities in a region. Four different types of transportations are available: *Flight*, *Bus*, *Taxi*, and *Train*.

6.1.2 Scenario – Demanding Collaboration

The following steps describe a scenario in the *Transport* example that may trigger a negotiation protocol to find a collaboration partner.

(1) A team's task is to monitor certain cities.

(2) A team member reports a transport status.

(3) An accident with some casualty is detected by team members and reported to a team leader.

(4) The team leader determines that a transport to a different city is required.

(5) The team leader finds out that the team can not transport by itself and needs collaboration.

(6) A negotiation protocol is fired to locate other team who can collaborate.

In the step (2), a member agent, *a1*, may send an event log message to a leader agent, *a0*, describing a context, "*Boston has a flight to New York*" with a timestamp, *"2008-07-16T13:24:56"*. It may be enveloped in 5W1H-based syntax utilizing OWL/XML or CELL as described in Chapter 5. The *Ont-Team* ontology defines an *Event* class with a set of properties including *who*, *what*, *whom*, *where*, and *when*. The log message is interpreted and stored into a whiteboard. A sample event log in CELL is as follows:

```
{ e1:Event |  \
    who={a1}, \
    what={s1:Log | subject={Boston | hasFlightTo={NewYork}}}, \
    whom={a0}, \
    where={Boston}, \
    when={:Time | timestamp="2008-07-16T13:24:56"} \
}
```

In the step (5), a team leader may find out if the transport can be accomplished from Boston to Miami by specifying a SQWRL query as following:

- City(?c1) ^ City(?c2) ^ hasTransTo(?c1, ?c2) ^ hasName(?c1, "Boston") ^ hasName(?c2, "Miami") -> sqwrl:select(?c1, ?c2) ^ sqwrl:count(?c2)

Upon returning a zero count, a negotiation protocol for collaboration may be triggered.

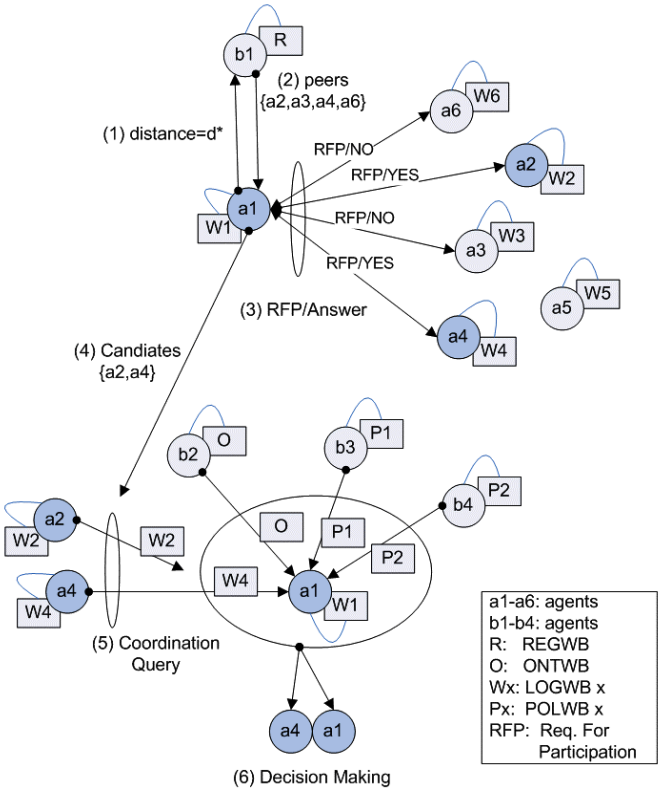

Figure 6-2 Negotiation Protocol

6.2. An Adaptive Policy-based Negotiation Protocol

This section introduces a negotiation protocol that relies on contextual information in whiteboards and adaptively copes with policies governing team situations. The protocol consists of four major steps: *Finding Peer Teams*, *Candidate Selection*, *Expand/Retry*, and *Decision Making*. Figure 6-2 outlines the steps and they are discussed further in this section.

6.2.1 Finding Peer Teams

In the negotiation protocol, finding peer teams may be accomplished by registry-based client-server or peer-to-peer protocols. The two approaches are similar in that both need to obtain the initial peer teams from the registry and send the participation requests. The difference between the approaches exists when expanding the request to find more peers.

The client-server approach relies on the requesting team's direct communication to all peer teams. It utilizes the registry to find out the peer teams in the farther distance whenever the request is expanded.

In the peer-to-peer approach, the requesting team only makes a direct communication to its local peers, and the peer teams propagates the request messages to their own peers. The registry lookup is minimal as it only searches each team's local peers. It sends a distance parameter as a part of a request message so that it limits the propagation within a

specified distance. There are a larger number of message exchanges unless each team node carefully keeps track of which request message has been propagated already.

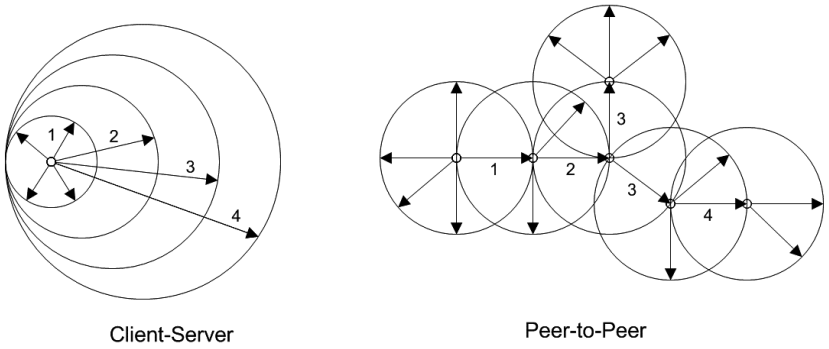

Figure 6-3 Expanding Peer teams

6.2.2 Candidate Selection

A team requiring coordination sends a Request-For-Participation (RFP) message to peer teams while a responding team replies back in case it can participate in the negotiation. The *RFP* message contains a request priority that is computed using a task and policies. For example, the organizational policies may define a rule giving a priority value for a specific task and its status. The following SWRL-based rule may be defined for the policies.

- Rescue(?k) ^ hasCasualty(?k, ?n) ^ (?n >= 1) ^ (?n < 3) -> hasPriority(?k, 6)

The responding team decides its participation depending upon the team situation and policies. As such, the protocol queries with multiple whiteboards including the team's

event logs and the organizational policies. It can compare the team's own task priority to the request priority, and identify if it has any urgent task. If there is no such prior or urgent task, it may reply a positive response.

- Task(?k) ^ hasPriority(?k, ?px) ^ (?px > ?req-priority) -> sqwrl:count(?k)
- Task(?k) ^ isUrgent(?k, true) -> sqwrl:count(?k)

What is an urgent task is defined by the organizational policies. For instance as a simple case, the policies may define that any task with a priority greater than a certain value is an urgent task:

- Task(?k) ^ hasPriority(?k, ?p) ^ (?p > 7) -> isUrgent(?k, true)

6.2.3 Expand and Retry

In case no candidate team is found, the participation request may be expanded by adjusting the distance and the request priority. With a bigger distance parameter in the peer finding, more teams are asked for the participation. A higher priority increases a potential for more peer teams to respond positively. The request priority may be increased based on a request number. A request number denotes the number of times that the same request is sent out. A larger request number means that the same request is sent out many times. As such, the request priority may be recomputed giving a higher value for a larger request number. Additionally, policies may define a rule to adjust a request priority.

6.2.4 Decision Making

The last step in the negotiation protocol is to decide a partner for collaboration. In the protocol, a decision relies on a coordination query, *coquery*, asking a specific context of interest using distributed context information from multiple whiteboards.

- S_{ij} <- coquery(q, <W1, W2>, <P>, . ..), for i >= 0 and j > 0

It executes a SQWRL query, *q*, with the references to LOGWBs, POLWBs, and ONTWB, and retrieves a result set, S_{ij}, in response to the context in question. Similar to a general database query, the result set, *Sij*, is a collection of vectors where each vector contains the attribute values that are projected by the query. The S_{ij} with the specific *i* and *j* denotes the value referenced by the j^{th} attribute in the i^{th} vector

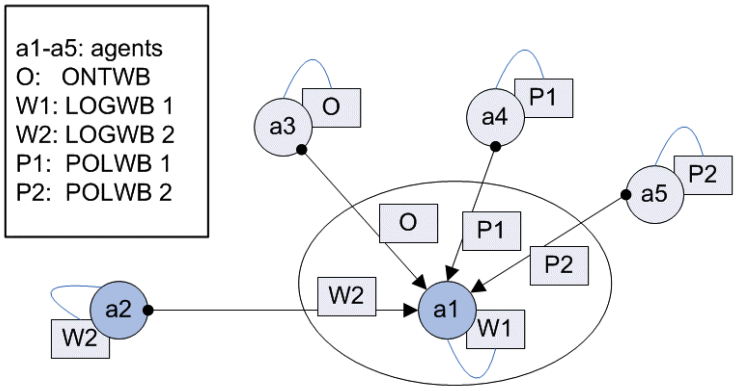

Figure 6-4 Distributed Context Sharing with Multiple Whiteboards

The protocol facilitates Decision Support Query (DSQ) that is a specific query predefined by the policies for a particular task. For instance, the policies may define a

DSQ, *transport.emergey.query1* for an emergency transport task, *k*. The protocol consults the POLWB to obtain this DSQ for the task and performs a *coquery*.

- DSQ <- get-DSQ(P, k)
- S_{ij} <- coquery(DSQ, <W1, W2>, <P>,..)

In the *Transport* example, the DSQ may be a context asking whether two teams can collaboratively accomplish the transport from Boston to Miami. While each team only knows about the regional transport information, the *coquery* may find if there is a route between two cities using inference rules. For instance, the following lines show a sample DSQ and an inference rule.

- City(?x) ^ City(?y) ^ hasTransTo(?x, ?y) ^ hasName(?x, "*c1*") ^ hasName(?y, "*c2*") -> sqwrl:select(?x, ?y)
- City(?x) ^ City(?y) ^ City(?z) ^ hasTransTo (?x, ?y) ^ hasTransTo(?y, ?z) -> hasTransTo(?x, ?z)

The protocol makes a query using the DSQ with Boston and Miami for *c1* and *c2*. A team may choose a candidate team for which the query returns a result, which means a route between two cities.

The negotiation protocol may further facilitate the coordination query with Decision Support Function (DSF) that computes a decision-score, or measure, describing how well the collaboration can be performed if two teams work together. It may be also retrieved from the POLWB. It is a numerical value computed using the result set, retrieved by the DSQ. In the *Transport* example, the DSQ may be a query to return a result set containing the vectors where each vector has two cities with an available transport.

The DSF may compute a decision-score by finding a shortest path among two cities, Boston and Miami, and giving a higher score for the path with less number cities. Alternatively, the DSQ may give a higher score for a larger number of paths between two cities. The former approach emphasizes efficiency while the latter favors reliability based on redundancy. The DSF may also facilitate a weighted sum combing both approaches. The following examples describe three variant DSFs that may compute the decision-score differently. The *c1* and *c2* are Boston and Miami in case of the *Transport* example.

- DSF1 : r1/count-cities(shortest-path(S_{ij}, <c1, c2>))
- DSF2 : r2*count-paths(all-paths(S_{ij}, <c1, c2>))
- DSF3 : *w1*DSF1 + *w2*DSF2,
  (*w1* and *w2* are weighting factors, and r1 and r2 are normalization factors)

There is no fixed or absolute DSQ and DSF computing the decision-score, but they are rather domain-specific relying on the policies governing an organization and situations. The negotiation protocol in the framework does not attempt to come up with the globally applicable DSQ and DSF, but it rather provides an adaptive protocol by separating the policies and rules from the actual protocol logic. The protocol only consults the POLWB to obtain the DSQ and the DSF.

The protocol may choose a candidate team with the highest decision-score as a collaboration partner, and confirms with the selected team. The minimum score to pass can be defined in the POLWB, similar to DSQ and DSF, and it may vary depending upon situations and policies.

- d = max{ DSF(S_{ij}(x), <c1, c2>) }
  where S_{ij}(x) <- coquery(DSQ, <W1, Wx>, <P>. …)
  for Tx is a team in the candidate set.

- (T1, Tx) is a collaboration partner if d >= d* such that d* <- get-MIN-DSF(P, k)

Under a circumstance where no partner team is found, the protocol retries by expanding peer teams and adjusting the request priority as described in the candidate selection.

Chapter 7 - Testing and Evaluation

This chapter presents testing and evaluations that are performed to demonstrate the operational conditions of the prototype. It discusses functional testing to verify the correctness of operations and a performance analysis to evaluate capabilities and potential issues. The chapter also includes the testing approach and environment as well as the analysis and discussions on the testing results.

7.1 Testing Environment and Approach

Two major aspects for evaluations are functional testing and a performance analysis on the prototype. Functional testing verifies the correctness of key components and their operations in the framework so that more complex system may be built on top of them. A performance analysis examines capabilities and identifies any potential issues for future improvements.

Testing is performed on three different machines running Windows operating systems. Functional testing is conducted on all of those machines to assert the correctness. For the performance analysis, a machine with the best physical configuration is selected to test the maximum capabilities of the prototype. The system configuration for testing is as follows:

- CPU: Intel Core 2, 2.4 GHz
- Memory: Physical - 4GB, Java VM - 1.5GB

- Languages: Java VM 1.6.0, Perl 5.8.7

Functional testing is also performed in three different levels in order to examine components and communications:

- Core whiteboard operations
- Service-level operations
- CELL scripts

Testing on core whiteboard operations examines the key APIs that may be directly integrated by a local application. Testing on service-level operations verifies the correctness of Web Services APIs that expose core whiteboard operations to a distributed environment. Testing with CELL scripts examines multiple components in an integration level that includes a translator and a whiteboard as well as underlying communication protocols.

Numerous test cases are defined in order to verify the functional correctness in different levels. Testing scripts are devised to use known input data and their outputs are asserted with the expected answers. In addition, testing also utilizes a Protégé-based visualization tool such that the populated whiteboard contents can be navigated and visually examined.

A performance analysis is conducted with programmatically-generated large datasets and a set of measures. In the experiment, a measure is the time that takes to perform a single or a set of operations on a dataset. It describes the significance to the performance

in the given dataset. Seven different measures are defined for the experiment and an analysis is described based on the results.

The performance with respect to different cluster sizes is also analyzed. A cluster in this experiment refers to a set of related data objects. It includes both directly and indirectly connected objects by properties. The performance of a query may vary significantly with a cluster size. A benchmark is performed with respect to several different cluster sizes, and its experiment results are discussed.

7.2 Functional Testing: Whiteboard Operations and Services

7.2.1 Procedure

A set of test cases are developed to examine core functionality of a whiteboard component. They are written with a Perl-based interface that operates with a Protégé runtime environment through an adaptor or Web Services. Those test cases assert the basic APIs for objects, properties, rules, and queries. Each test case invokes a series of whiteboard APIs that are devised to produce outputs for verification.

7.2.2 Test Cases for Core Whiteboard Operations

The first group of test cases verifies the basic APIs in order to get and set objects, properties, and relationships. It can be utilized to build more complex whiteboard contents. Figure 7-1 shows a skeleton of testing routines where operations on different properties such as single, multiple, and object values tested. The actual test cases cover more comprehensive list of property types and values.

```
$o1 = $wb->createWbObject('AllPropTypes', "o1");
$o1->set('hasStringProp', "str1");
$o1->set('hasStringPropMultiple', ["str2", "str3"]);

$o2 = $wb->createWbObject('AllPropTypes', "o2");
$o2->set('hasObjectProp', $o1);

$str  = $o2->get('hasObjectProp')->get('hasStringProp');
$o2->delete();

$list = $wb->findWbObjects("AllPropTypes");
```

Figure 7-1 Test Case for Objects and Properties

The second group of test cases demonstrates the construction of a higher-level interface by combining a set of basic APIs. Because a whiteboard relies on ontologies that are highly expressive and flexible, its APIs can be combined in diverse ways as shown in Figure 7-2.

```
sub createCity
{
  my ($self, $wb, $objName, $city, $state, $region) = @_;

  my $obj = $wb->addWbObject('City', $objName);
     $obj->set('hasCityName', $city);
     $obj->set('hasStateName', $state);
     $obj->set('hasRegionName', $region);

  return $obj;
}

sub setHasFlightTo
{
  my ($from, $to) = @_;
  $from->set('hasFlightTo', $to);
}

$boston = $self->createCity($wb, "Boston", "Boston", "MA", "East Coast");
$ny = $self->createCity($wb, "NewYork", "New York", "NY", "East Coast");
setHasFlightTo($boston, $ny);
```

Figure 7-2 Test Case for Building Blocks

Figure 7-3 depicts the test cases for rules and queries. The whiteboard APIs supports a direct use of the SWRL and SQWRL syntax in a convenient way. A method, queryBySpecRules2, facilitates a SQWRL query and SWRL rules to select a set of connected cities by any transport.

```
$rules = [
 "City(?c1) ^ City(?c2) ^ City(?c3) ^ hasTransTo(?c1,?c2)
  ^ hasTransTo(?c2,?c3) -> hasTransTo(?c1,?c3)",
 "City(?c1) ^ City(?c2) ^ hasFlightTo(?c1,?c2) -> hasTransTo(?c1,?c2)",
 "City(?c1) ^ City(?c2) ^ hasTrainTo(?c1,?c2) -> hasTransTo(?c1,?c2)",
 "City(?c1) ^ City(?c2) ^ hasBusTo(?c1,?c2) -> hasTransTo(?c1,?c2)",
 "City(?c1) ^ City(?c2) ^ hasTaxiTo(?c1,?c2) -> hasTransTo(?c1,?c2)"
];

$query =
  "City(?c1) ^ City(?c2) ^ hasTransTo(?c1,?c2) ^ hasCityName(?c1,?from)
   ^ hasCityName(?c2,?to) ^ swrlb:notEqual(?from,?to)
   -> sqwrl:select(?from,?to) ^ sqwrl:orderBy(?from)";

$result = $wb->queryBySpecRules2($query, $rules);
```

Figure 7-3 Test Case for Rules and Queries

A *coquery* is a key API that asks a specific context of interest using distributed information from multiple whiteboards. A test case in Figure 7-4 examines the *coquery* with three different whiteboards that contain contexts and rules.

```
populate($wb1);
populate($wb2);
populate($pol, $rules);

$result = $wb1-> coquery(
              $query,
              [ $wb1->getWbOntUri(), $wb2->getWbOntUri() ],
              [ $pol->getWbOntUri() ]
          );
```

Figure 7-4 Test Case for coquery

7.2.3 Test Cases for Service-level Operations

The service-level operations are the Web Services APIs that are based on core whiteboard APIs. As such, a Web Services test client is created to consume those service-based APIs, and the similar test cases in the previous section are reused for the verification.

7.3 Functional Testing: CELL Scripts

This section discusses functional testing based on CELL scripts. Interpreting a CELL script internally triggers a sequence of steps using a translator and whiteboard operations. A CELL-based test script can verify the functional correctness of multiple components and protocols.

7.3.1 Procedure

A set of CELL scripts is defined to test objects, event logs, rules, and queries. Test scripts are devised to produce outputs to verify the correctness. Testing a CELL script facilitates the scripting environment, supported by the whiteboard prototype. The prototype can run as a standalone CELL interpreter that translates a CELL script and performs whiteboard operations. Each CELL script is invoked with this standalone interpreter for testing.

7.3.2 Test Cases Using CELL Scripts

Figure 7-5 is a CELL script to examine the line-based specification of object creations. Logically, it is analogous to the test cases for building blocks in Figure 7-2.

```
{Boston:City | hasCityName=Boston, hasStateName=MA, \
               hasRegionName="East Coast"}
{NewYork:City | hasCityName="New York", \
                hasStateName=NY,\hasRegionName="East Coast" }
```

Figure 7-5 CELL script for Object Creation

Figure 7-6 shows a CELL script to test an event log. It represents a single line of 5W1H-based event log as discussed in Chapter 5. The event log is translated and its contexts are stored in a whiteboard.

```
{ e1:Event |  \
    hasWho={a1}, \
    hasWhat={ s1:Log |  \
                hasSubject={Boston | hasFlightTo={NewYork}} \
            }, \
    hasWhom={a0}, \
    hasWhere={Boston}, \
    hasWhen={ :Time | hasTimestamp="2008-07-16T13:24:56"} \
}
```

Figure 7-6 CELL Script for Event Log

Figures 7-7 and 7-8 show CELL scripts for examining rules and queries. SWRL rules and SQWRL queries can be directly embedded within a CELL script and executed with the coquery using multiple whiteboards.

```
%{ policy1.rule1 | City(?c1) ^ City(?c2) ^ City(?c3) \
    ^ hasTransTo(?c1,?c2) ^ hasTransTo(?c2,?c3) -> hasTransTo(?c1,?c3) }
%{ policy1.rule2 | City(?c1) ^ City(?c2) ^ hasFlightTo(?c1,?c2) \
    -> hasTransTo(?c1,?c2) }
%{ policy1.rule3 | City(?c1) ^ City(?c2) ^ hasTrainTo(?c1,?c2) \
    -> hasTransTo(?c1,?c2) }
%{ policy1.rule4 | City(?c1) ^ City(?c2) ^ hasBusTo(?c1,?c2) \
    -> hasTransTo(?c1,?c2) }
%{ policy1.rule5 | City(?c1) ^ City(?c2) ^ hasTaxiTo(?c1,?c2)\
    -> hasTransTo(?c1,?c2) }

%{ policy1.query1 | City(?c1) ^ City(?c2) ^ hasTransTo(?c1,?c2) \
    ^ hasCityName(?c1,?from) ^ hasCityName(?c2,?to) \
    ^ swrlb:notEqual(?from,?to)\
    -> sqwrl:select(?from,?to) ^ sqwrl:orderBy(?from) }
```

Figure 7-7 CELL Script for Rules and Queries

```
>{ coquery %{policy1.query1} \
      [http://wb1/url, http://wb2/url] \
      [http://policy/url] }
```

Figure 7-8 CELL Script for coquery

7.3.3 Whiteboard Content Visualizations

With a visualization tool, data contents and structures can be navigated and visually examined. The whiteboard contents are ontologies and their instances that consist of classes, objects, and properties. As such, ontology visualization tools can be also reused for testing and verifying valid structures and contents. Figures from 7-9 to 7-14 are the screenshots of whiteboard contents that are visualized with several different Protégé-based plug-in tools.

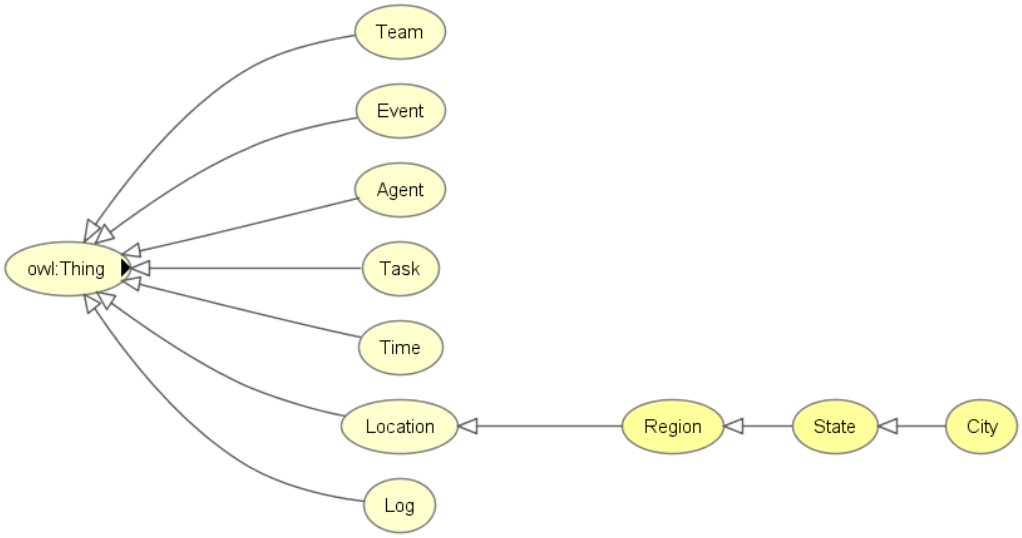

Figure 7-9 Sample A - Whiteboard Ontology Visualization by Protégé/Owlviz

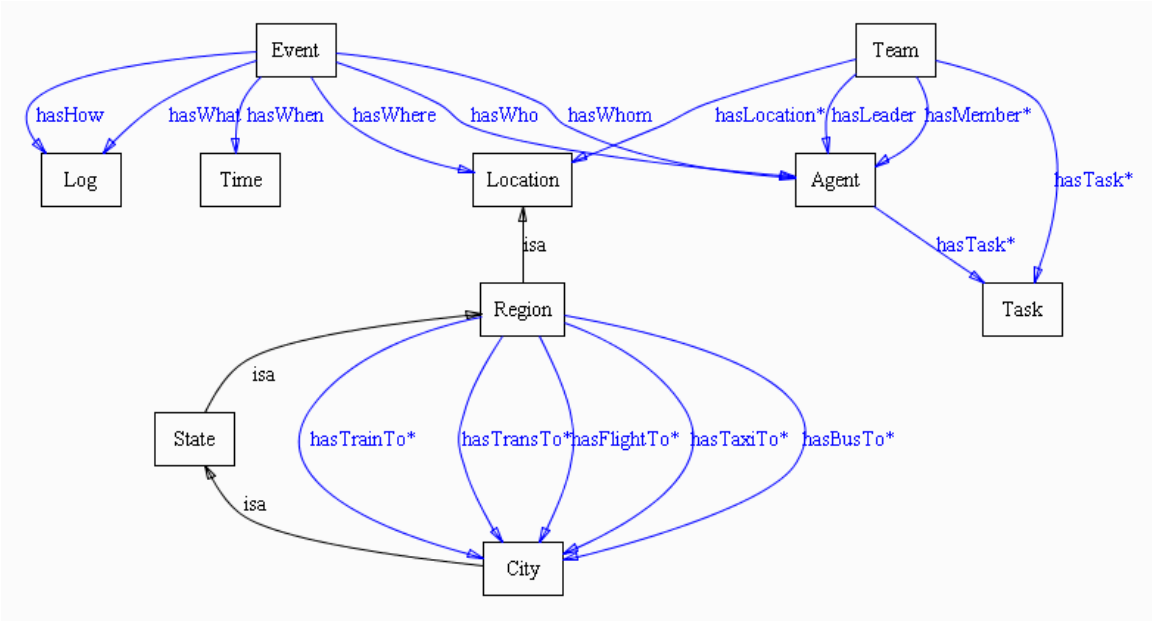

Figure 7-10 Sample B - Whiteboard Ontology Visualization by Protégé/Ontoviz

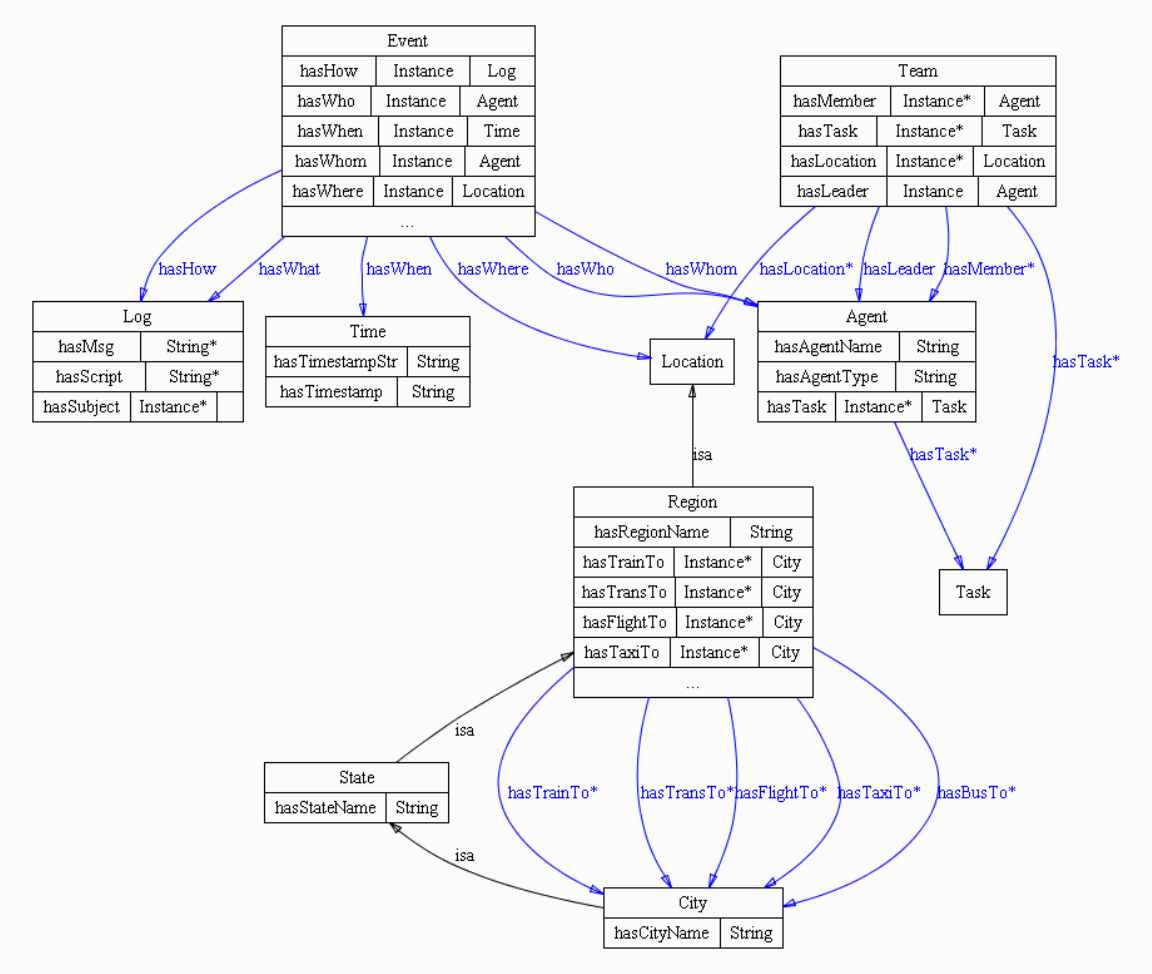

Figure 7-11 Sample C - Whiteboard Ontology Visualization by Protégé/Ontoviz

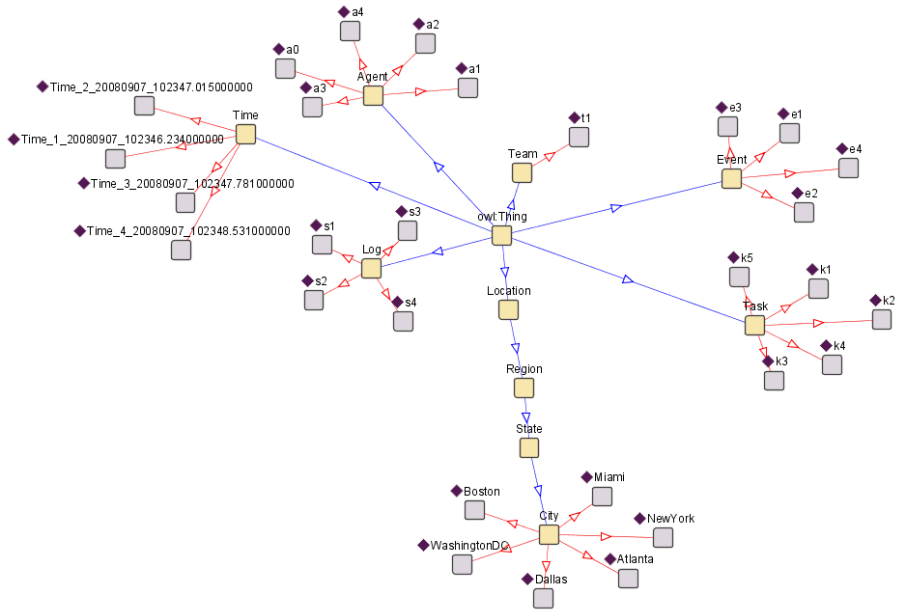

Figure 7-12 Sample D - Whiteboard Contents Visualization by Protégé/Jambalaya

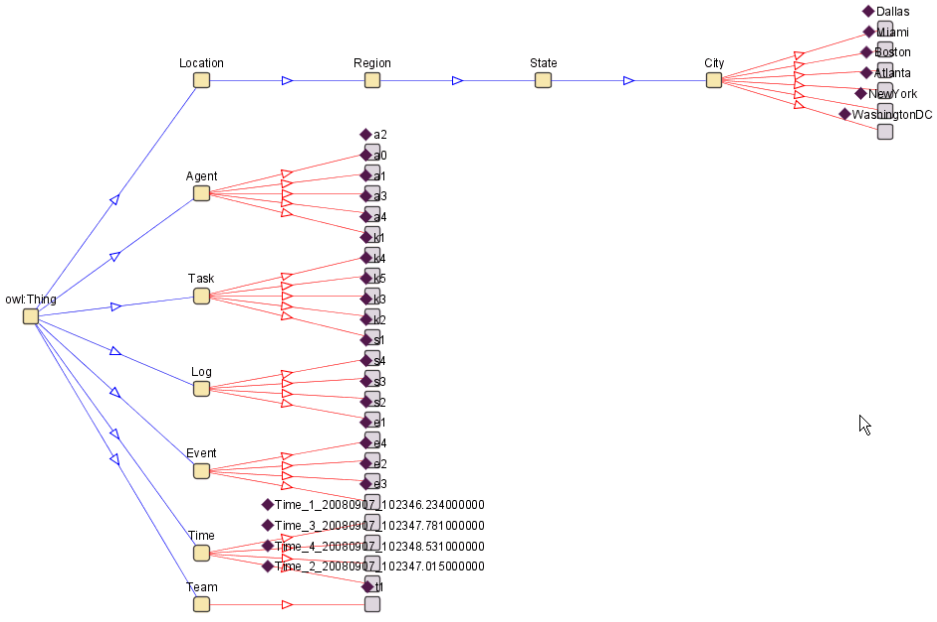

Figure 7-13 Sample E - Whiteboard Content Visualization by Protégé/Jambalaya

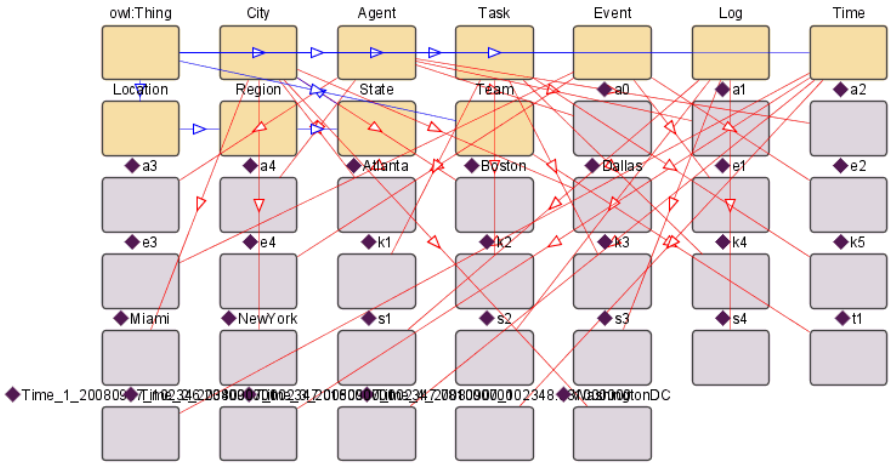

Figure 7-14 Sample F - Whiteboard Content Visualization by Protégé/Jambalaya

7.4 Performance Analysis

This section presents a performance analysis of the prototype. The prototype is data-intensive that is based on ontological information as well as rule-based querying and reasoning. Its performance is significantly affected by a set of underlying technologies, adapted to the prototype implementation. It relies on Protégé programming libraries and the runtime that is bridged to a Jess rule engine. An investigation of an optimal implementation is beyond the scope of this research. However, a performance study on the current prototype may examine its capabilities and address any potential issues for future enhancements.

7.4.1 Procedure

Five datasets are programmatically generated up to 100,000 data objects for a transport example in Chapter 6. Each dataset is a collection of city objects with several properties including a randomly-selected transport type that connects to other city objects. A connection to other city is set to either an adjacent or a randomly-chosen city, but every forth or sixth city is forced to be a sink city without any outgoing transport. It is due to the practical limitation of the underlying runtime and rule engine. The system deteriorates with excessive memory consumptions when a query is performed on a dataset with a large cluster. A cluster in this experiment refers to a set of related city objects. In the experiment, each dataset is constructed with a cluster size of approximately three or five with possible random variations.

Several performance measures are defined to test the whiteboard capabilities. The measures are follows:

- T_TOTAL.

  The total number of seconds that takes to execute all measures for a dataset.

- T_QUERY.

  The number of seconds that takes to query with reasoning for any transport from a source city to any reachable destination cities.

- T_FIND.

  The number of seconds that takes to locate a named city object in a dataset.

- T_CREATE.

  The number of seconds that takes to create all city objects and set the transport connection properties in a dataset.

- T_CITIES

  The number of seconds that takes to create all city objects in a dataset.

- T_TRANS

  The number of seconds that takes to set the transport connection properties of all cities objects in a dataset.

- T_DELETE.

  The number of seconds that takes to delete all city objects in a dataset

7.4.2 Result of Performance Measures

Figures 7-15 and 7-16 show the result of the performance measures on a dataset of the size up to 100,000 objects in a whiteboard. T_CREATE is the most expensive where the time increases linearly with respect to the data size. T_FIND is constant regardless of the data size. T_DELETE is less affected by the data size. T_QUERY is faster than T_CREATE, but it consumes excessive memories, reaching almost to 1.6GB. The system deteriorates when the size of dataset is beyond 100,000 objects. Creating a context log is logically equivalent to creating a set of related data objects in a whiteboard. Note that T_CREATE is the time that takes to create entire data objects in a dataset. In practice, creating a context log is much faster as it only creates a few set of related data objects.

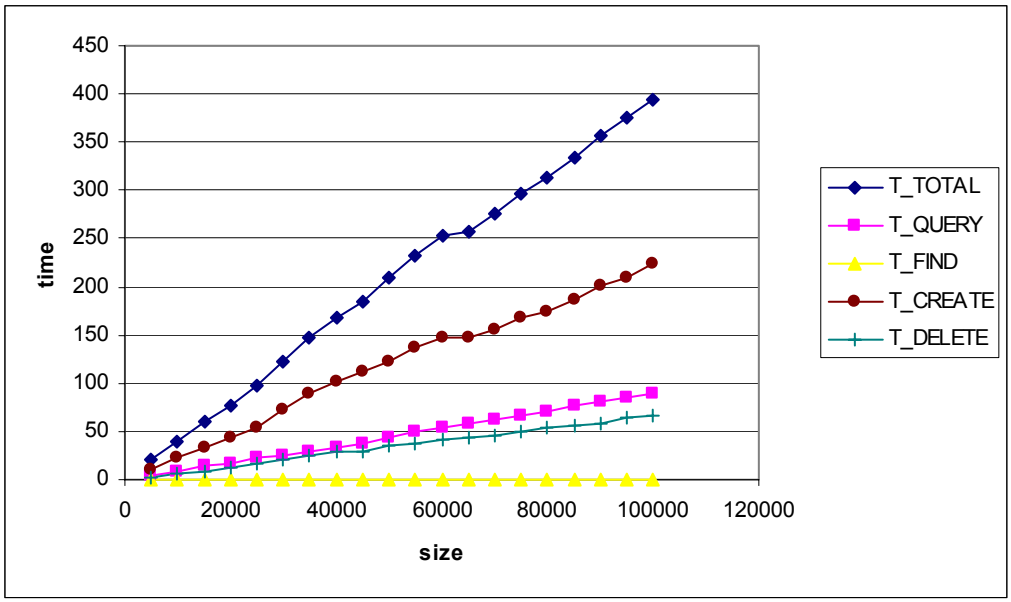

Figure 7-15 Performance Measures - Every 5000 Objects

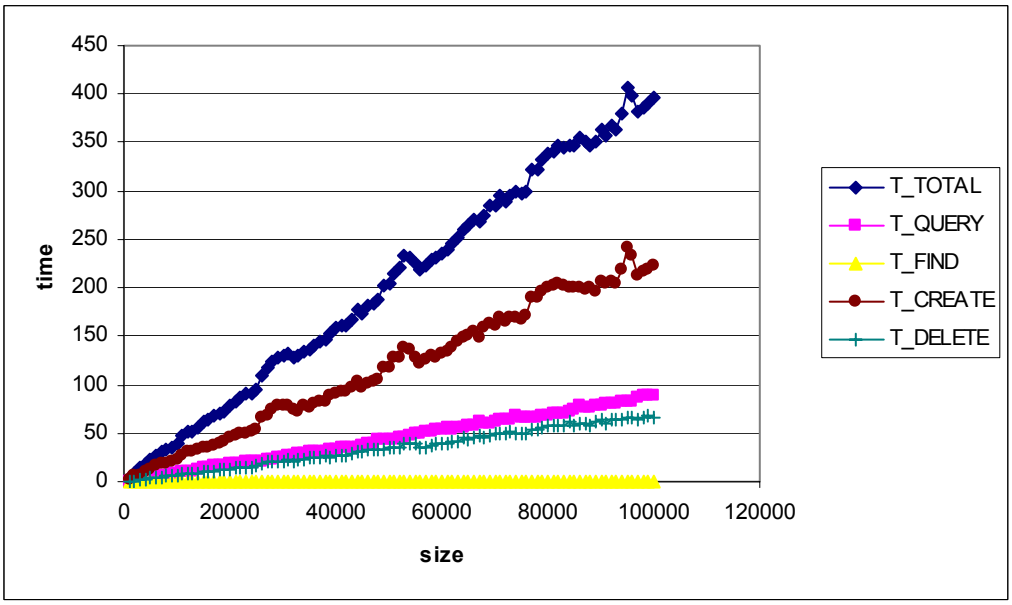

Figure 7-16 Performance Measures - Every 1000 Objects

Figure 7-17 shows the performance result of the T_CREATE measure on a dataset with 100,000 objects. T_CREATE is the addition of T_CITIES and T_TRANS. T_CITIES, which is the time for creating data objects without any relationships, appears to be the most influential to the creation time. T_TRANS is the measure for setting a relationship property, and is less influential.

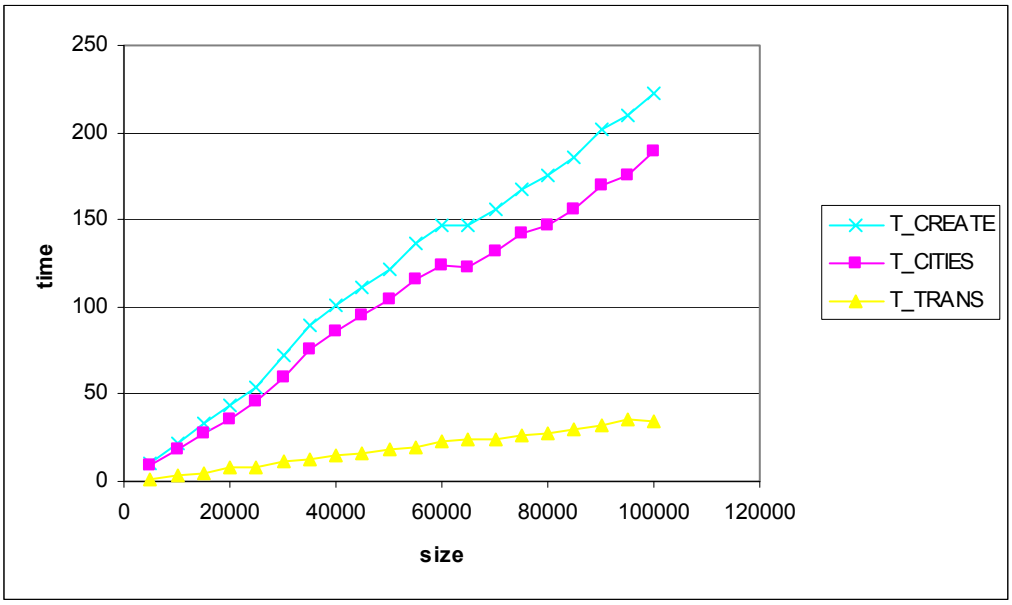

Figure 7-17 T_CREATE Performance

7.4.3 Result of Variant Cluster Size

Figure 7-18 shows the T_QUERY performance on four different datasets with 50,000 objects. Each dataset is programmatically generated with a different cluster size.

Datasets, T05 and T07, have a fixed cluster size of three and five, respectively. T06 and T08 are the variations of T05 and T07 where a connection to other city is randomized. T_QUERY on T05 performs efficient while that on T08 are slower. In T08, the system starts deteriorating when the dataset size is 50,000 or more. A randomized connection to other city can result in a large cluster, possibly with a long chain of related data objects. It appears the performance of the rule engine is significantly influenced by this cluster size. T_QUERY on T06 and T07 shows a similar performance.

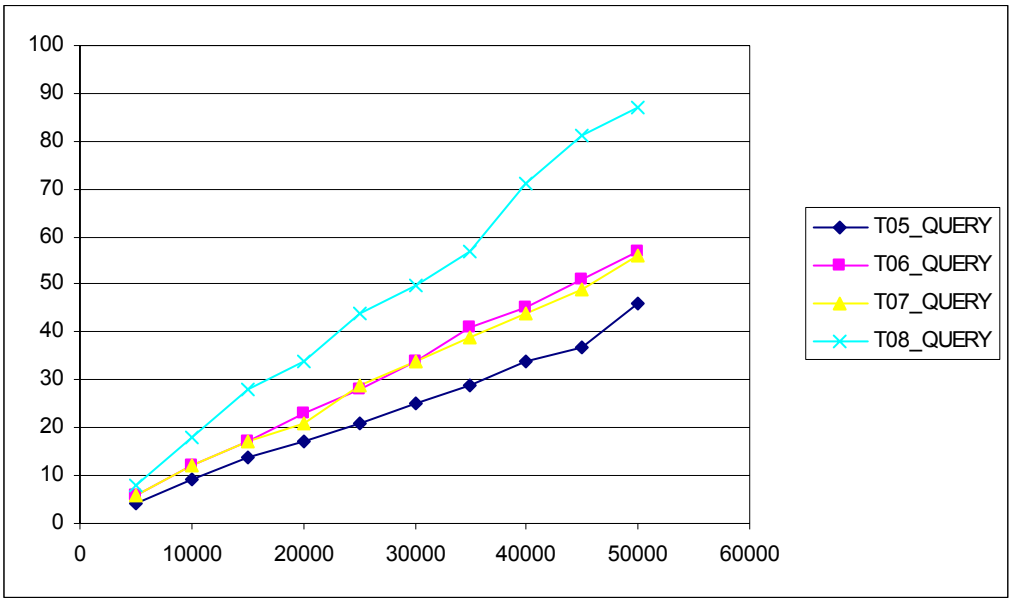

Figure 7-18 Performance of Different Cluster Sizes

7.5 Discussion

All testing results in Sections 7.2 and 7.3 verify the functional correctness of components and operations. Testing is performed on three different machines running Windows XP and Vista. All test cases successfully produce the expected answers.

Although testing is performed on Windows only, the prototype is expected to run with other platforms, languages, and protocols. The prototype is built with Perl and Java that can run on many other platforms. A service-based APIs enable any applications written in different languages to integrate the whiteboard prototype. Although it is not explicitly discussed earlier, the delivery of a CELL-based message using Syslog is promising. It can be successfully transmitted and logged to a whiteboard. All of those running options can enable more experiments under different environments in the future.

The analysis in Section 7.4 shows that the performance of a whiteboard is significantly affected by the amount of data and system memories. With a reasonable data size, the whiteboard runs efficiently while it deteriorates gradually with a larger size. It may be due to the default configuration of the Protégé runtime. It facilitates a file-based backend storage that is loaded into memories for running. It may produce a higher performance because of in-memory data managements, but are bound to the memory size. An alternative configuration based on a database may overcome the limitation. A future research on this issue may increase the scalability of the whiteboard.

The performance is also bound to a cluster size within a dataset. A cluster in the experiment refers to a set of related objects. With a large cluster, the performance degrades. It appears the rule engine deteriorates with a poor performance and excessive

memory consumptions when a dataset contains a large cluster. The current Protégé configuration at the moment when this research is conducted only supports a bridge-based Jess rule engine. For the future research, it may be worthwhile to experiment with an alternative option.

Chapter 8 - Conclusions and Future Work

8.1 Conclusions

This dissertation presents a whiteboard-based computational framework that supports dynamic team coordination in a distributed environment. The framework consists of four principle components to address the key issues: representation, communication, log transfer syntax, and negotiation. A whiteboard component facilitates Semantic Web to represent contexts and rules. A whiteboard agent provides flexible yet interoperable communication. Context-aware Event Log Language (CELL) specifies the log transfer syntax of fully-structured event logs. Situation/Policy-Aware Negotiation Protocol (SPAN) allows a team to find a collaboration partner, adaptively coping with dynamic situations and policies.

A prototype is implemented as a proof of concept for the proposed approach. It includes the core features of components that are developed using several platforms and technologies including Java, Perl, Protégé programming libraries, and Jess. The prototype is evaluated in terms of functional testing to verify the correctness of operations and a performance analysis to examine capabilities and potential issues. Prototyping and testing demonstrate the proposed approach is feasible.

The major contributions of the dissertation include a number of novel approaches and new technological developments. The framework streamlines several technologies and architectures into a common computational foundation that may be potentially adapted to

any problem domain. It includes a new whiteboard model that enhances a blackboard-based approach to an information sharing component, supporting awareness of situations and policies. A hybrid approach in communication architectures allows flexible interactions among distributed applications and agents. A scripting language, CELL, and its translator are developed for the specification of fully-structured event logs that may be easily delivered as data streams or lines. A negotiation protocol, SPAN, devises policy-based coordination among teams under dynamically-changing conditions.

The other contributions of the research are the utilizations of ontologies and Semantic Web technologies. A new generic context ontology, Ont-Team, is defined to support core concepts for team coordination, and it may be extended to a problem-specific environment. The research devises an approach to facilitate Semantic Web as a service-based component, providing an easier integration environment for distributed applications. Associated with a CELL translator, it also supports a stream-based communication to Semantic Web. Lastly, the research introduces a whiteboard prototype that is a simple-to-use tool relying on Semantic Web. While Semantic Web is often considered as a complex technology, it may provide an experimental environment for researchers and technology explorers.

8.2 Future Work

In addition to the current development of the framework, there is much work yet to improve and explorer further to demonstrate the utility of the framework in a distributed computing environment. This section outlines future research directions.

8.2.1 Improvements

The use cases with more complex team structures and coordination scenarios can be developed. The research includes a simple example for team situations although it addresses fundamental issues for coordination. A more complex example may find any hidden deficiency or other issues for improvements.

A generic context ontology, Ont-Team, may be enhanced with richer concepts and properties. In the research, it only facilitates a minimum set of definition. While it may be extended upon additional demands, a richer common ontology may increase potential context-level interoperability and simplify the extension to a problem-specific environment.

The framework components may be optimized in a fine-grained level. For instance, the candidate selection step in the SPAN protocol sends a simple Request For Participation (RFP) message to peer teams. With an additional hint context that may describe the purpose of a collaboration request, the protocol may filter the number of candidate teams in an early stage and reduce the processing time for a decision.

A feasibility study can be conducted for alternative technologies to the backend storage. A whiteboard component is data-intensive and its performance may be significantly affected by the amount of data. The current prototype relies on the default configuration in the Protégé runtime system. The backend storage is file-based where querying and reasoning take place in memory after loading. A performance analysis with

a different backend technology such as Sesame [138] or database may suggest a more scalable configuration for the component.

8.2.2 Integration and Interoperability

The Syslog integration with the framework can be fully implemented and tested with existing facilities. Syslog is a commonly-used standard protocol to deliver log messages from distributed applications or facilities. A preliminary experiment shows the integration is promising. It may enhance applications to deliver fully-structured semantic logs on top of the existing logging infrastructure.

Interoperability between the framework and Foundation for Intelligent Physical Agents (FIPA) [109] agents may be investigated. FIPA is the standard organization for agent and multi-agent systems that specifies Agent Communication Language (ACL) [110] and related standards. The framework includes a whiteboard agent that is built with open standards and environments such as Web Services and Semantic Web. Consequently, the interoperability with FIPA agents is an interesting problem to explorer as it may enable FIPA agents to utilize the framework, or vice versa.

References

[1] A. K. Dey, "Understanding and Using Context," *Personal and Ubiquitous Computing Journal*, **5**, pp. 4-7, Feb 2001.

[2] E. D. Mynatt, "The Writing on the Wall," in *Proceedings of the 7th IFIP Conference on Human-Computer Interaction (INTERACT'99)*, M.A. Sasse and C. Johnson, eds., IOS Press, Edinburgh, UK, 1999, pp. 196-204.

[3] T. Igarashi, W. K. Edwards, A. LaMarca, and E. D. Mynatt, "An Architecture for Pen-based Interaction on Electronic Whiteboards," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, Palermo, Italy, May 2000, pp.68-75.

[4] Y. Xiao, S. Faraj, C. Mackenzie, M. E. Mills, and A. Norcio, "Coordination Processes and Awareness Support in Dynamic Work Environments," Final Report, National Science Foundation Award IIS-9900406, 2002.

[5] J. Hwang and S. Rotenstreich, "A Policy-Based Coordination Model for Team Collaboration," in *Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009)*, Baltimore, MD, USA, May 18-22, 2009, pp. 131-138.

[6] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, pp. 34-43, May 2001.

[7] W3C, "W3C Semantic Web Activity," http://www.w3.org/2001/sw. Site accessed on October 19, 2009.

[8] N. Shadbolt , T. Berners-Lee, and W. Hall, "The Semantic Web Revisited," *IEEE Intelligent Systems*, **21**, pp.96-101, May 2006,

[9] D. J. Weitzner, J. Hendler, T. Berners-Lee, and D. Connolly, "Creating a Policy-Aware Web: Discretionary, Rule-based Access for the World Wide Web," in *Web and Information Security*, E. Ferrari and B. Thuraisingham, eds., IRM Press, Hershey, PA, 2006.

[10] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," W3C Working Group Note, 11 February 2004, http://www.w3.org/TR/ws-arch. Site accessed on April 16, 2009.

[11] H. Haas and A. Brown, "Web Services Glossary," W3C Working Group Note, 11 February 2004, http://www.w3.org/TR/ws-gloss. Site accessed on April 15, 2009.

[12] S. Vinoski, "Serendipitous reuse," *IEEE Internet Computing*, **12**(1), pp. 84-87, 2008.

[13] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, CA, USA, 2000.

[14] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture," in *Proceedings of the 22nd international conference on Software engineering*, 2000, pp.407-416.

[15] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture," *ACM Transactions on Internet Technology (TOIT)*, **2**, pp. 115-150, May 2002.

[16] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision," in *Proceedings of the 17th international conference on World Wide Web Conference*, 2008, pp. 805-814.

[17] D. Garlan and M. Shaw, "An Introduction to Software Architecture," Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CS-94-166, 1994.

[18] C. V. Ramamoorthy and H. F. Li, "Pipeline Architecture," *ACM Computing Surveys (CSUR)*, **9**, pp. 61-102, March 1977.

[19] T. W. Malone and K. Crowston, "What is coordination theory and how can it help design cooperative work systems," in *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, Los Angeles, CA, USA, 1990, pp. 357-370.

[20] T. W. Malone and K. Crowston, "The Interdisciplinary Study of Coordination," *ACM Computing Surveys*, **26**(1), pp. 87-119, 1994.

[21] J. A. Espinosa, R. E. Kraut, J. F. Lerch, S. A. Slaughter, J. D. Herbsleb, and A. Mockus, "Shared Mental Models and Coordination in Large-Scale, Distributed Software Development," in *Proceedings of the International Conference on Information Systems (ICIS 2001)*, New Orleans, LA, USA, December, 2001, pp. 513-518.

[22] J. A. Espinosa, K. M. Carley, R. E. Kraut, S. A. Slaughter, F. J. Lerch, and S. Fussell, "The Effect of Task Knowledge Similarity and Distribution on Asynchronous Team Coordination and Performance: Empirical Evidence from Decision Teams," in *Second Information Systems Cognitive Research Exchange (ISCoRE) workshop*, Barcelona, Spain. December, 2002.

[23] J. A. Cannon-Bowers, E. Salas, and S. A. Converse, "Cognitive psychology and team training: Training shared mental models of complex systems," *Human Factors Society Bulletin*, **33**(12), pp. 1-4, 1991.

[24] M. Orasanu, "Shared mental models and crew decision making," CSL Report No. 46, Princeton. NJ: Princeton University, Cognitive Science Laboratory, 1990.

[25] E. E. Entin and D. Serfaty, "Adaptive team coordination," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, **41**(2), pp. 312-325, 1999.

[26] Y. Xiao, C. Lasome, J. Moss, C. Mackenzie, and S. Faraj, "Cognitive Properties of a Whiteboard: A Case Study in a Trauma Centre," in *Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work*, Bonn, Germany, 2001. pp. 259-278.

[27] C. L. Plasters, F. Jacob Seagull, and Y. Xiao, "Coordination Challenges in Operating-Room Management: An In-Depth Field Study," in *Proceedings of 2003 American Medical Informatics Association, Annual Symposium*, 2003, pp. 524-528.

[28] E. D. Mynatt, T. Igarashi, W. K. Edward, and A. LaMarca, "Flatland: New dimensions in office whiteboards," in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 1999)*, Pittsburgh, Pennsylvania, New York, ACM Press, pp. 346-353.

[29] Z. Shae, B. Tseng, and W. H. Leung, "Immersive Whiteboard Collaborative System," in *Annals of Software Engineering*, **12**(1), pp.193-212, December 2001.

[30] Z. Zhang and L. He, "Whiteboard It! Convert Whiteboard Content into an Electronic Document," Microsoft Research, August 2002, http://research.microsoft.com/en-us/um/people/zhang/whiteboardit/whiteboardit.pdf. Site accessed on March 13, 2009.

[31] D. D. Corkill, K. Q. Gallagher, and P. M. Johnson, "Achieving Flexibility, Efficiency, and Generality in Blackboard Architectures," in *Proceedings of the 6th National Conference on A.I.*, Seattle, Washington, 1987, pp. 18-23.

[32] D. D. Corkill, “Blackboard Systems,” *AI Expert*, **6**(9), pp. 40-47, 1991.

[33] D. D. Corkill, “Collaborating Software: Blackboard and Multi-Agent Systems & the Future,” in *Proceedings of the International Lisp Conference*, New York, NY, October 2003.

[34] O. Selfridge, “Pandemonium: A Paradigm for Learning,” in *Proceedings of Symposium on the Mechanization of Thought Processes*, 1959, pp. 511-529.

[35] R. Engelmore and A. Morgan, *Blackboard Systems*, Addison-Wesley, 1986.

[36] R. Adler, “Blackboard Systems,” *The Encyclopedia of Artificial Intelligence*, S. C. Shapiro eds., pp. 110-116. New York: Wiley Interscience, 1992.

[37] K. R. Thorisson, T. List, C. Pennock, and J. DiPirro, “Whiteboards: Scheduling Blackboards for Semantic Routing of Messages & Streams,” in *AAAI-05 Workshop on Modular Construction of Human-Like Intelligence*, 2005.

[38] C. Boitet and M. Seligman, “The "whiteboard" architecture: a way to integrate heterogeneous components of NLP systems,” in *Proceedings of COLING’94*, Kyoto, Japan, 1994, pp. 426-430.

[39] T. Morimoto, M. Suzuki, T. Takezawa, G. Kikui, M. Nagata, and M. Tnmokiyo, “A Spoken Language Translation System: SL-TRANS2,” in Pro*ceedings of the 14th Conference on Computational Linguistics (COLING'92)*, Nantes, France, Vol. **3**, 1992, pp.1048-1052.

[40] T. Morimoto, T. Takezawa, F. Yato, S. Sagayama, T. Tashiro, M. Nagata, and A. Kurematsu, “ATR's speech translation system: ASURA,” in *3rd European Conference on Speech Communication and Technology (EUROSPEECH '93)*, Berlin, Germany September, 1993, pp1291-1294.

[41] L. D. Erman and V. R. Lesser, “The Hearsay-II Speech Understanding System: A Tutorial,” *Trends in Speech Recognition*, W.A. Lea, eds., Prcntice-Hall, 1978.

[42] R. Reddy, “Machine Models of Speech Perception,” *Perception and Production of Fluent Speech*, R. Cole, eds., Erlbaum, NJ, pp. 215-242, 1980.

[43] J. Wolf and W. Woods, “The HWIM Speech Understanding System,” *Trends in Speech Recognition*, W. A. Lea, eds., Prentice-Hall, pp. 316-339, 1980.

[44] L. Cahill, C. Doran, R. Evans, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper, "In Search of a Reference Architecture for NLG systems," in *Proceedings of EWNLG*, 1999.

[45] L. Cahill, C. Doran, R. Evans, R. Kibble, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper, "Enabling Resource Sharing in Language Generation: An Abstract Reference Architecture," in *Proceedings of LREC*, 2000.

[46] P. Neophytou, A. Labrinidis, and P. K. Chrysanthis, "WhiteBoard P2P: A Peer-to-Peer Reliable Data Dissemination Infrastructure for Collaborative Applications," Department of Computer Science, University of Pittsburgh, 2006. http://www.cs.pitt.edu/~panickos/publications/conference/wbp2p-isyc06.pdf. Site accessed on June 20, 2009.

[47] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *IEEE Workshop on Mobile Computing Systems and Applications,* Santa Cruz, CA, US, 1994.

[48] B. Schilit and M. Theimer, "Disseminating Active Map Information to Mobile Hosts," *IEEE Network*, **8**(5), pp 22-32, 1994.

[49] J. Pascoe, "Adding Generic Contextual Capabilities to Wearable Computers," in *Proceedings of 2nd International Symposium on Wearable Computers*, 1998. pp 92-99.

[50] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey," in Workshop on Advanced Context Modeling, Reasoning and Management, UbiComp 2004, The 6th International Conference on Ubiquitous Computing, 2004, pp. 33-40.

[51] A. K. Dey, D. Salber, and G. D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," Human-Computer Interaction Journal, **16**(2-4), pp. 97-166, 2001.

[52] T. Gu, H.Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004, pp. 270-275.

[53] W3C, "Composite Capabilities/Preferences Profile (CC/PP)," http://www.w3.org/Mobile/CCPP. Site accessed on March 22, 2009.

[54] S. Buchholz, T. Hamann, and G. Hubsch, "Comprehensive Structured Context Profiles (CSCP): Design and Experiences," in *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, March 2004.

[55] W3C, Extensible Markup Language (XML), http://www.w3.org/XML. Site accessed on April 16, 2009.

[56] W3C, XML Schema, http://www.w3.org/XML/Schema. Site version: December 24, 2008.

[57] D. Connolly, "Standard Generalized Markup Language," W3C, November 1995, http://www.w3.org/MarkUp/SGML. Site version March 26, 2004.

[58] P. S. Chen, "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transaction on Database Systems*, **1**, pp. 9.36, March 1976.

[59] Unified Modeling Language (UML), Object Management Group(OMG), http://uml.org. Site version: October 13, 2009.

[60] Object Management Group (OMG), http://www.omg.org. Site accessed on October 19, 2009.

[61] T. A. Halpin, Object Role Modeling (ORM), http://www.orm.net. Site accessed on October 19, 2009.

[62] T. Halpin, "ORM2," in *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, R. Meersman, et al. eds., Springer LNCS 3762, 2005, pp 676-87.

[63] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling context information in pervasive computing systems," in *Proceedings of 1st International Conference on Pervasive Computing, Zurich, Switzerland, 2002*, F. Mattern and M. Naghshineh, eds., LNCS 2414, Springer, pp. 167-180.

[64] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Generating Context Management Infrastructure from High-Level Context Models," in *Proceedings of the 4th International Conference on Mobile Data Management (MDM'2003)*, Melbourne, Australia, January 2003, pp. 1-6.

[65] J. D. Ullman and J. Widom, *A first course in database systems*. New Jersey: Prentice Hall, Inc, 1997.

[66] A. Silberschatz, H. F. Korth, and S. Sudershan, *Database System Concepts, 3rd edition*. New York: McGraw-Hill, Inc., 1998.

[67] M. Kifer, A. Bernstein, and P. M. Lewis, *Database Systems: An Application Oriented Approach, Compete Version, 2nd Edition*. Boston: Addison-Wesley, 2005.

[68] B. Bouzy and T. Cazenave, "Using the Object Oriented Paradigm to Model Context in Computer Go," in *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'97)*, Rio, Brazil, 1997, pp. 279-289.

[69] B. B. Stroustrup, *The C++ Programming Language, Second Edition*, MA:Addison-Wesley Publishing Company, 1991.

[70] J. McCarthy, "Notes on formalizing contexts," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, San Mateo, CA, 1993, R. Bajcsy, eds., Morgan Kaufmann, pp. 555-560.

[71] J. McCarthy and S. Buvac, "Formalizing context (expanded notes)," *Computing Natural Language, CSLI Lecture Notes*, **81**, pp. 13-50, 1998.

[72] C. Ghidini and F. Giunchiglia, "Local models semantics, or contextual reasoning = locality + compatibility," *Artificial Intelligence*, **127**(2), pp. 221-259, 2001.

[73] V. Akman and M. Surav, "The use of situation theory in context modeling," *Computational Intelligence*, **13**(3), pp. 427-438, 1997.

[74] J. Barwise and J. Perry, "Situations and Attitudes," *The Journal of Philosophy*, **78**(11), pp. 668-691, 1981.

[75] L. R. Khan, "Ontology-based Information Selection," Ph.D. dissertation, University of Southern California, CA, USA, 2000.

[76] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, **5**, pp. 199-220, 1993.

[77] T. R. Gruber, "A Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Knowledge Acquisition*, **5**, pp. 199-220, 1993.

[78] R. Krummenacher and T. Strang, "Ontology-Based Context-Modeling," in *Third Workshop on Context Awareness for Proactive Systems (CAPS'07)*, 2007.

[79] E. J. Ko, H. J. Lee, and J. W. Lee, "Ontology-Based Context Modeling and Reasoning for U-HealthCare," in *IEICE TRANSACTIONS on Information and Systems*, **90**, pp. 1262-1270, 2007.

[80] N. Belhanafi, C. Taconet, and G. Bernard, "CAMidO, A Context-Aware Middleware based on Ontology meta-model," in *Workshop on Context Awareness for Proactive Systems*, June, 2005, pp. 93-103.

[81] J. R. Hilera and F. Ruiz, "Ontologies in Ubiquitous Computing," in *Proceedings of First International Conference on Ubiquitous Computing (ICUC)*, Alcala de Henares, Spain, June 7-9, 2006, pp. 109-116.

[82] P. Floreen, M. Przybilski, P. Nurmi, J. Koolwaaij, A. Tarlano, M. Wagner, M. Luther, F. Bataille, M. Boussard, B. Mrohs, and S. Lau, "Towards a Context Management Framework for MobiLife," in *14th IST Mobile & Communications Summit*, Dresden, Germany, 2005.

[83] K. C. Lee, J. H. Kim, J. H. Lee, and K. M. Lee, "Implementation of Ontology Based Context-Awareness Framework for Ubiquitous Environment" in *Proceedings of International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, Seoul, Korea, April 2007, pp. 278-282.

[84] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology-Based Context Modeling and Reasoning using OWL," in *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004, pp. 18-22.

[85] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-features.

[86] H. Chen, F. Perich, T. Finin, and A. Joshi, "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications," in *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2004, pp. 258-267.

[87] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," W3C Recommendation, 10 February, 2004, http://www.w3.org/TR/owl-ref.

[88] M-R. Koivunen and E. Miller, "W3C Semantic Web Activity," W3C, 2001, http://www.w3.org/2001/12/semweb-fin/w3csw.

[89] D. Connolly, F. van Harmelen, I. Horrocks, D. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "DAML+OIL (March 2001) Reference Description," W3C, 2001, http://www.w3.org/TR/daml+oil-reference.

[90] F. Manola and E. Miller, "RDF Primer," W3C Recommendation, 10 February 2004, http://www.w3.org/TR/rdf-primer.

[91] T. Berners-Lee, "Primer: Getting into RDF & Semantic Web using N3," W3C, October 2000, http://www.w3.org/2000/10/swap/Primer. Site version: August 16, 2005.

[92] H. Chen, T. Finin, and A. Joshi, "An Ontology for Context-Aware Pervasive Computing Environments," *The Knowledge Engineering Review*, **18**(3), pp. 197 - 207, September 2003.

[93] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C Member Submission, 21 May 2004, http://www.w3.org/Submission/SWRL.

[94] The Rule Markup Initiative, RuleML, http://www.ruleml.org. Site version: August 11, 2009.

[95] ProtegeWiki, "SQWRL, Semantic Query-enhanced Web Rule Language," 2008, http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL. Site version: August 6, 2009.

[96] Stanford Center for Biomedical Informatics Research, Protégé, http://protege.stanford.edu. Site accessed on October 19, 2009.

[97] T. Berners-Lee, "CWM," W3C, October 2000, http://www.w3.org/2000/10/swap/doc/cwm.html. Site version: February 5, 2008.

[98] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System Of Patterns*. West Sussex, England: John Wiley & Sons Ltd., 1996.

[99] D. Garlan, R. Allen, and J. Ockerbloom, "Exploiting style in architectural design environments," in *Proceedings of the 2nd ACM SIGSOFT symposium on Foundations of software engineering*, 1994, pp175-188.

[100] C. Lonvick, “The BSD syslog Protocol,” IETF RFC 3164, August 2001, http://tools.ietf.org/html/rfc3164.

[101] OMG, “CORBA/IIOP Specification,” Object Management Group, http://www.omg.org/technology/documents/formal/corba_iiop.htm. Site version: April 27, 2009.

[102] S. Vinoski, “Corba: Integrating diverse applications within distributed heterogeneous environments,” *IEEE Communications Magazine*, **14**(2), 1997.

[103] S. Vinoski, “Web Services Interaction Models, Part 1: Current Practice,” *IEEE Internet Computing*, **6**(3), pp. 89- 91, May, 2002.

[104] S. Vinoski, “Putting the "Web" into Web Services: Interaction Models, Part 2,” *IEEE Internet Computing*, **6**(4), pp. 90- 92, July, 2002.

[105] N. R. Jennings, “On agent-based software engineering,” *Artificial Intelligence*, **117**(2), pp.277-296, March 2000.

[106] M. Wooldridge and P. Ciancarini, “Agent-Oriented Software Engineering: The State of the Art,” in *First International Workshop on Agent-Oriented Software Engineering (AOSE 2000)*, Limerick, Ireland, June 10, 2000, LNCS 1957, Springer, 2001, pp. 55-82.

[107] N. R. Jennings, “An Agent-based Approach For Building Complex Software Systems,” *Communications of the ACM*, **44**(4), pp. 35-41, April, 2001.

[108] M. Wooldridge, “Agent-based software engineering,” *IEE Proceedings on Software Engineering*, **144**(1), pp. 26-37, 1997.

[109] The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org. Site accessed on October 19, 2009.

[110] FIPA, “FIPA Agent Communication Language Specifications,” http://www.fipa.org/repository/aclspecs.php3. Site accessed on April 22, 2009.

[111] B. Schiemann and U. Schreiber, “OWL DL as a FIPA ACL content language,” in *Proceedings of the Workshop on Formal Ontology for Communicating Agents (FOCA)*, 18th European Summer School of Language, Logic and Information, University of Malaga, Spain, July 2006, pp. 73-80.

[112] M. Pasha, H. F. Ahmad, A. Ali, and H. Suguri, "Semantic Grid: Interoperability Between OWL and FIPA SL," in *Proceedings of 9th Pacific Rim International Workshop on Multi-Agents (PRIMA 2006)*, Guilin, China, August 7-8, 2006, LNCS 4088, Springer, pp. 714-720.

[113] X. T. Nguyen and R. Kowalczyk, "WS2JADE: Integrating Web Service with Jade," in *Proceedings of AAMAS 2007 International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2007)*, Honolulu, HI, USA, May 14, 2007, LNCS 4504, Springer, 2007, pp. 147-159.

[114] D. Greenwood, M. Lyell, A. Mallya, and H. Suguri, "The IEEE FIPA Approach to Integrating Software Agents," in *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007) - Industrial Track*, Hawaii, USA, 2007.

[115] I. Foster, N. R. Jennings, and C. Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, New York, 2004, pp. 8-15.

[116] ITU-T, "Languages for Telecommunication Systems," *International Telecommunication, Union*, http://www.itu.int/ITU-T/studygroups/com17/languages. Site accessed on April 18, 2009.

[117] B. S. Kaliski, "A Layman's Guide to a Subset of ASN.1, BER, and DER," An RSA Laboratories Technical Note, Revised November 1, 1993, http://luca.ntop.org/Teaching/Appunti/asn1.html.

[118] International Standard Organization, http://www.iso.org. Site accessed on October 19, 2009.

[119] International Telecommunication Untion, http://www.itu.int. Site accessed on October 19, 2009.

[120] M. Hori, J. Euzenat, and P. F. Patel-Schneider, "OWL web ontology language XML presentation syntax," W3C Note, 11 June 2003, http://www.w3.org/TR/owl-xmlsyntax.

[121] XIOP - XML Corba Environment-Specific Inter-ORB Protocol, 2000, http://xml.coverpages.org/xiop.html. Site version: June 30, 2000.

[122] Introducing JSON, http://json.org. Site accessed on October 19, 2009.

[123] D. Crockford, “The application/json Media Type for JavaScript Object Notation (JSON)”, IETF RFC 4627, July 2006, http://www.ietf.org/rfc/rfc4627.txt.

[124] C. C. Evans, “YAML Ain't Markup Language,” http://yaml.org. Site accessed on October 19, 2009.

[125] O. Ben-Kiki, C. Evans, and I. dot Net, “YAML Ain't Markup Language, Version 1.2,” YAML.org, Working Draft 2008-05-11, 2008.

[126] F. Zulkernine, P. Martin, C. Craddock, and K. Wilson, “A Policy-based Middleware for Web Services SLA Negotiation,” in *Proceedings of the IEEE International Conference on Web Services (ICWS’08), Industry Track*, September 2008.

[127] W3C, “Web Services Policies 1.2 - Framework (WS-Policy),” April 2006, http://www.w3.org/Submission/WS-Policy.

[128] P. C. K. Hung, H. Li, and J.-J. Jeng, “WS-Negotiation: An overview of research issues,” in *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS'04),* Big Island, Hawaii, January 2004, pp. 33-42.

[129] S. Y. W. Su, C. Huang, J. Hammer, Y. Huang, H. Li, L. Wang, Y. Liu, C. Pluempitiwiriyawej, M. Lee, and H. Lam, “An Internet-based Negotiation Server for ecommerce,” *the VLDB Journal*, **10**(1), pp. 72-90, August 2001.

[130] C. Huang, “A Web-based negotiation server for supporting electronic commerce,” Ph.D. dissertation, University of Florida, USA, 2000.

[131] H. Gimpel, H. Ludwig, A. Dan, and B. Kearney, “PANDA: Specifying Policies for Automated Negotiations of Service Contracts,” in *Proceedings of International Conference on Service Oriented Computing (ICSOC 2003)*, Trento, Italy, 2003, pp. 287-302.

[132] M. Khedr and A. Karmouch, “Negotiating Context Information in Context-Aware Systems,” *IEEE Intelligent Systems*, **19**(6), pp.21-29, November 2004.

[133] Y. Al Ridhawi and A. Karmouch, “Ontology-based negotiation protocol and context-level agreements,” in *IET 4th International Conference on Intelligent Environments (IE08)*, Seattle, WA, July 2008, pp.1-8.

[134] G. Dimitoglou, "Business Rule Computation In Distributed Organizations," Ph.D. dissertation, The George Washington University, Washington DC, USA, 2003.

[135] Sandia National Laboratories in Livermore, Jess, http://herzberg.ca.sandia.gov. Site version: November 11, 2008.

[136] W. B. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structures & Algorithms*. New Jersey: Prentice Hall, Inc., 1992.

[137] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann, 2001.

[138] OpenRDF.org, http://www.openrdf.org. Site accessed on October 19, 2009.