Semantic Parameterization: A Process for Modeling Domain Descriptions

TRAVIS D. BREAUX, ANNIE I. ANTÓN and JON DOYLE
North Carolina State University

Software engineers must systematically account for the broad scope of environmental behavior that is described in non-functional requirements and includes the coordinated actions of stakeholders and software systems. The Inquiry Cycle Model (ICM) provides engineers with a strategy to acquire and refine these requirements by having domain experts answer six questions: who, what, where, when, how and why. Goal-based requirements engineering has led to the formalization of requirements to answer the ICM questions about *when*, *how* and *why* goals are achieved, maintained or avoided. In this paper, we present a systematic process called Semantic Parameterization for expressing natural language domain descriptions of goals as specifications in Description Logic. The formalization of goals in Description Logic allows engineers to automate inquiries using *who*, *what* and *where* questions, completing the formalization of the ICM questions. The contributions of this approach include new theory to conceptually compare and disambiguate goal specifications that enables querying goals and organizing goals into specialization hierarchies. The artifacts in the process include a dictionary that aligns the domain lexicon with unique concepts, distinguishing between synonyms and polysemes, and several natural language patterns that aid engineers in mapping common domain descriptions to formal specifications. Semantic Parameterization has been empirically validated in three case studies on policy and regulatory descriptions that govern information systems in the finance and health-care domains.

Categories and Subject Descriptors: D.2.1 [**Requirements/ Specifications**]: Languages and Methodologies

General Terms: Documentation, Standardization, Human Factors

Additional Key Words and Phrases: Natural Language, domain knowledge, formal specification, Description Logic

1. INTRODUCTION

To improve the validity of software requirements and designs, requirements and software engineers must consider the role of the software system in the broader con-

Authors' addresses: T.D. Breaux, A.I. Antón, J. Doyle, Department of Computer Science, North Carolina State University, 890 Oval Drive, Campus Box 8206, Raleigh, NC 27695-8206, email: {tdbreaux,aianton,jon_doyle}@ncsu.edu.
This work was supported in part by National Science Foundation ITR Grant: *Encoding Rights, Permissions and Obligations: Privacy Policy Specification and Compliance (NSF 032-5269)*; ITR CyberTrust Grant: *Policy-Driven Framework for Online Privacy Protection (NSF 043-0166)*; the IBM PhD Fellowship; and CERIAS at Purdue University.

text of stakeholder goals. These goals focus the requirements engineering effort on satisfying specific environmental states that include non-functional requirements. These states are either achieved, maintained or avoided by actors who perform actions within the environment [van Lamsweerde 2001]. Developing formal descriptions of both the system, actors and the environment will reduce stakeholder misconceptions about the role of the system and expose tacit interactions within the environment that are necessary to ensure broader requirements coverage. In the discussion that follows, we use the term *domain* to refer to a set of conceptually related environments and *domain description* to refer to a stakeholder's conceptualization or transcription of a specific domain problem in natural language [Jackson and Zave 1993].

Goal-based methods exist to extract goals from domain descriptions [Antón 1996] and to formalize goals for automated analysis [Dardenne et al. 1993; Fuxman et al. 2004]. The Goal-Based Analysis Method (GBRAM) provides engineers with guidelines and heuristics to acquire semi-formal instances of actors, goals and constraints [Antón 1996]. The GBRAM supports the Inquiry Cycle Model (ICM), which drives requirements elicitation by asking who, what, where, when, how and why questions to domain experts [Potts et al. 1994]. The model provides an interactive structure by which stakeholders and analysts iteratively elaborate upon requirements through expression, discussion and commitment, contributing to the refinement of requirements. Inquiry helps analysts and stakeholders identify what information is missing and which assumptions are pending. In GBRAM, the answers to ICM questions yield new artifacts in the form of goals or domain descriptions that refine or

condition previously identified goals [Antón 1996]. By supporting the ICM during formalization, engineers encourage the continued involvement of the domain expert helping to disambiguate and refine formal models that are derived from domain descriptions.

The Knowledge Acquisition in AutOmated Specification (KAOS) [Dardenne et al. 1993] and Tropos [Fuxman et al. 2004] methods provide engineers a means to formalize several goal operators (achieve, maintain, avoid) using linear-time temporal logic (LTL) and many important goal concepts using conceptual graphs (CG). In the goal refinement process, the LTL representations formalize inquiries about *when* goals are satisfied whereas the CGs formalize inquires about *why* and *how* goals are satisfied. Goal-related questions about *who* performs an action, *what* objects are acted upon, *where* actions are performed require the expressive power to reason about classes of individuals (e.g., instances) using concepts and roles. In knowledge representation, concepts and roles are defined abstractly in domain-level or intensional knowledge using sub-class or specialization relationships. These definitions are independent from instance-level or extensional knowledge, which describes specific problems within the domain by assigning individuals to pre-defined concepts and roles. Although KAOS distinguishes between domain and instance-level knowledge [Dardenne et al. 1993], the class relationships are exclusively defined using CGs and cannot adequately be expressed and reasoned about in LTL. On the other hand, Tropos uses no sub-classing or specialization of structures and thus the domain- and instance-levels are not distinguished in the formalization. Although GBRAM directs stakeholders to ask all of the ICM-related questions, the lack of

goal formalization in first-order predicate logic precludes the automation of the inquiry process using logical inference.

To address this limitation in KAOS, Tropos and GBRAM, we consider a fragment of first-order logic called Description Logic (DL) that is used to represent and reason about knowledge at the domain- and instance-levels [Baader et al. 2002]. The basic inference mechanism in DL is called *subsumption* and is used to infer whether or not one concept is contained in the set of subclasses of another concept and, inversely, to infer whether one concept is more general than another. By assigning individuals to concepts, DL can be used to reason about the similarity of individuals through shared concepts at different levels of abstraction. Unfortunately, it is a non-trivial task to map goal structures from KAOS, Tropos and GBRAM into DL. For example, in KAOS, Tropos and GBRAM, each goal is represented by a semi-formal natural language statement that consists of a verb followed by a phrase. The phrase is often represented as a single proposition that prohibits asking more detailed who, what and where questions using logical inference. In KAOS [Dardenne et al. 1993] for example, the goal "Achieve BorrowerRequestSatisfied" implies several tacit activities in the goal phrase, including: the intent of an actor (the borrower) to borrow; the request of the borrower; and the perception by some actor of satisfying the borrower's request. This goal also has several ambiguities that are identifiable by asking the right ICM questions, including: *what* the borrower is borrowing; *what* the borrower is requesting; *who* perceives the satisfaction (e.g., the borrower or the lender); and *how* the satisfaction is measured. These questions are typically raised during the Inquiry Cycle with the answers supplied by a domain expert. Because

the phrase is informal, unstructured and compounded into a single proposition, these questions can only be identified and answered by an engineer who recognizes these ambiguities in the label given to this proposition. Similar to KAOS, both GBRAM and Tropos yield goals with phrases that are also informal and inaccessible to automated inquiry [Antón et al. 2004; Fuxman et al. 2004].

We build upon this prior work and present a process called Semantic Parameterization that engineers can use to systematically map natural language domain descriptions into DL expressions. The process can be used by engineers and domain experts to formalize otherwise vaguely defined sources of knowledge and distinguish knowledge about the domain (e.g., concepts and roles) from knowledge about problems (e.g., individuals and interactions) in the domain. The rigorously derived DL expressions support asking *who*, *what*, *where*, *why* and *how* questions about instances in the domain relevant to the Inquiry Cycle Model. These questions extend the role of the domain expert in disambiguating and refining formal models of domain descriptions. The process assists in the identification of four types of ambiguity: (1) synonymy (same-meaning) and (2) polysemy (multiple meaning) by mapping the lexicon to unique concepts in a dictionary; (3) anaphora (backward references) or cataphora (forward references) by applying NL phrase heuristics to domain descriptions; and (4) under-specifications or omissions by applying NL patterns to domain descriptions. We provide a standard procedure for deriving generalizable NL patterns from domain descriptions. Finally, we summarize several NL patterns that were successfully employed to formalize frequently recurring requirements phrases in a substantial body of work across three case studies.

The remainder of this paper is organized as follows: in Section 2 we present related work, focusing on methods to map domain descriptions into structured and semi-structured goals; in Section 3 we present the Semantic Parameterization process; in Section 4 we present the NL patterns for modeling goals and requirements; in Section 5 we present the empirical validation from three case studies; with our discussion and summary in Section 6, where we show the results of two formal techniques, enabled by our approach, to analyze goals.

2. RELATED WORK

Several researchers have recognized the need to better align natural language requirements and formal models to: prevent the loss of original context [Potts 1997]; incorporate knowledge about the system environment [Jackson 1997; van Lamsweerde 2000; Mylopoulos et al. 1997; Nuseibeh and Easterbrook 2000]; improve traceability [Ramesh and Jarke 2001]; apply formal analysis to requirements concepts [van Lamsweerde 2000; Nuseibeh and Easterbrook 2000]; and reduce ambiguous terminology [Gause and Weinberg 1989; Jackson 1997; Denger et al. 2003; Wasson et al. 2003; Wasson 2006]. In this paper, we focus on the process to model natural language descriptions of systems and their environments. Therefore, we first review the role of ontology in modeling domain knowledge before reviewing two popular methods for acquiring formal specifications from natural language descriptions: controlled languages [Cregan et al. 2007; Konrad and Cheng 2005; Kaljurand and Fuchs 2007; Smith et al. 2002] and standard lexicons [Cysneiros and Leite 2004; Kaindl 1996; Overmyer et al. 2001; Wasson et al. 2003; Wasson 2006]. We conclude by reviewing a few types of ambiguities identified by Gause

and Weinberg [Gause and Weinberg 1989]. Throughout this overview, we discuss how Semantic Parameterization extends or complements the related work.

In knowledge representation, an ontology defines terms in classification hierarchies in which conceptually more abstract terms appear higher in the hierarchy. An *upper ontology* describes the most abstract terms that are more frequently shared across multiple domains. In the KAOS framework, the meta-level concepts are most likely to appear in an upper ontology. The IEEE Standard Upper Ontology Working Group (IEEE P1600.1) was established to produce a standard upper ontology to support computer applications. Example material from different upper ontologies can be found in Cyc [Matuszek et al. 2006], DOLCE [Gangemi et al. 2002], the Suggested Upper Merged Ontology (SUMO) [Niles and Pease 2001], and WordNet [Fellbaum 1998]. There is a debate concerning the existence, feasibility and relevance of an upper ontology to coordinate shared knowledge across multiple domains and users [Welty 2002]. Because upper ontologies tend to lead to extraneous issues beyond the scope of a single software system, we have developed our process in such a way that the requirements models are limited to the meaning that domain experts assign directly to domain descriptions. In large software projects, this sharp focus is often essential to success of the project. Consequently, we take the approach that experienced domain experts must properly align separate ontologies when that need arises.

Controlled languages, which comprise a subset of natural language, have been developed in requirements engineering [Breaux and Antón 2005b; 2005a; Konrad and Cheng 2005; Denger et al. 2003; Smith et al. 2002] and artificial intelligence

[Chen 1983; Kaljurand and Fuchs 2007; Cregan et al. 2007] to reduce ambiguity and inconsistency in natural language specifications. Smith et al. describe the PROPEL tool that uses disciplined natural language (DNL) templates to capture requirements [Smith et al. 2002]. The templates permit a limited number of concise, highly-structured phrases that correspond to formal properties used in finite state automata. Konrad and Cheng employ a structured English grammar with special operators tailored to the specification of real-time properties [Konrad and Cheng 2005]. Templates and structured grammars require the engineer to focus the domain description in a manner consistent with pre-defined operators in a formal method. Denger et al. use natural language patterns to capture conditional, temporal and functional requirements statements [Denger et al. 2003]. Similar to templates and grammars, we provide natural language patterns in Section 4 that are intended to help engineers restate goal descriptions into Description Logic expressions. These expressions extend prior work by allowing engineers to formally reason about the classification and composition of goal concepts.

In artificial intelligence, we highlight three approaches to map a subset of the English language to entity-relationship (ER) models [Chen 1983] and Description Logic, including Attempo Controlled English (ACE) by Kaljarund et al. [Kaljurand and Fuchs 2007] and Computer-Processable ENGlish (PENG) by Cregan et al. [Cregan et al. 2007]. Peter Chen proposed eleven rules to manually extract entities, relations and attributes in ER models from English sentences [Chen 1983]. ER models require engineers to decide whether a "thing" is an entity or a relation between entities, an ambiguity we call the node-edge problem, which we discuss in

Section 3.1. We identify new rules in addition to the rules identified by Chen and present these new rules in Section 4. ACE [Kaljurand and Fuchs 2007] and PENG [Cregan et al. 2007] include two approaches to align a subset of natural language (English) with a formal semantics in DL. An important issue that arises during this alignment includes words with an anaphoric or cataphoric function, such as English pronouns and definite articles (e.g., this, that, the), that refer the reader to a particular thing or individual described in a prior or subsequent context, respectively. Our approach extends their work by elaborating upon a method to formally distinguish all shared individuals using anaphoric and cataphoric references and to unambiguously map these individuals into assertions in DL. Because ACE has been mapped to DL using "artificial" examples that lack coherence and relevance to a particular domain [Kaljurand and Fuchs 2006; 2007], their results are still preliminary and anecdotal. In Section 5, we present empirical validation of our approach in three case studies over two domains.

In requirements engineering, it is common practice to standardize the natural language vocabulary using a lexicon or dictionary. Antón et al. applied the Goal-Based Requirements Analysis Method (GBRAM) [Antón 1996] to policies to extract goals that begin with a verb followed by a goal phrase [Antón and Earp 2004; Antón et al. 2004]. In GBRAM, these verbs are standardized in a shared lexicon to avoid redundant goals. Overmyer et al. describe the Linguistic Assistant for Domain Analysis (LIDA) tool that maintains a list of words acquired from natural language documents; the words are used to identify classes and attributes in the UML [Overmyer et al. 2001]. Similarly, Kaindl shows how to identify binary

relationships between nouns in natural language definitions and map them to new classes [Kaindl 1996]. Wasson et al. employ a domain map to facilitate effective communication between domain experts and engineers [Wasson et al. 2003; Wasson 2006]. The domain map contains technical words classified into hierarchies of super-ordinate and sub-ordinate terms and is used to identify ambiguous terms based on their commonality and domain-specific interpretation. Cysneiros and Leite model non-functional, natural language requirements in the UML using class, sequence and collaboration diagrams [Cysneiros and Leite 2004]. Their approach uses a Language Extended Lexicon (LEL) to codify the natural language vocabulary in terms of denotations and connotations. In our approach, we employ a dictionary that maps words in a lexicon to their meanings in an ontology expressed in Description Logic. The dictionary is used to resolve three types of ambiguity: (1) synonymy (same meaning) and (2) polysemy (multiple meanings) at both the conceptual and real-world knowledge levels; and (3) under-specifications or omissions that are implied by relations to other concepts (i.e., the word "patient" implies a relationship to a hospital, doctor, or some other agent that provides treatment to the patient.)

Ambiguity can appear in requirements specifications that use natural language representations. Gause and Weinberg describe three types of ambiguity: (1) missing requirements, such as missing constraints on properties of things; (2) ambiguous words, including adjectives such as "small" or "inexpensive;" and (3) new words that are introduced in requirements statements that did not appear in the original domain description [Gause and Weinberg 1989]. Our process supports detecting some ambiguities in all three of these categories. For example, the process exposes

missing constraints on the properties of actions and goals. These properties are assigned a formal interpretation in Description Logic that is presented in Table IV. Furthermore, the dictionary separately distinguishes adjectives that appear in domain descriptions and can be used to detect value-laden terms such as "small" and "inexpensive" in requirements models. While our process does not prevent engineers from introducing new words during the formalization of domain descriptions, the process does separate new words from words that appear in the domain description by using a meta-model, which is described in Section 3.2. This separation supports scrutinizing new words to identify potential ambiguities in domain descriptions. Finally, a fourth type of ambiguity called *syntactic ambiguity* includes sentence phrases that may be attributed to more than one noun or phrase in the sentence. For example, the sentence "The health plan notifies patients with e-mail" is syntactically ambiguous because it is unclear whether "with e-mail" is attributed to the notification (e.g., use e-mail to notify patients) or to the patients (e.g., patients who have e-mail are notified). This example is formalization in DL in Section 3.3. The risk of misinterpreting this type of ambiguity can lead to incorrectly specifying the requirement.

3. SEMANTIC PARAMETERIZATION

Semantic Parameterization is a process to support engineers who map natural language domain descriptions to models expressed in first-order predicate logic for the purpose of performing automated reasoning and analysis. The process was developed to support the following three goals:

(1) Provide a reference system similar to natural language that allows stakeholders

to make ambiguous statements about systems and that systematically detects and resolves such ambiguities. This system is realized through semi-automated procedures that combine tools, knowledge bases and user feedback. The dictionary that is based on Description Logic and presented in this section supports this goal by documenting the many formal interpretations of domain words.

(2) Provide automated support for placing natural language-like inquiries across collections of requirements that answer *who*, *what*, *where*, *when*, *how* and *why* questions [Potts et al. 1994]. The query enables comparing requirements, which is necessary to build more advanced requirements analysis techniques such as organizing requirements, identifying conflicts, etc. We validated this design goal in two case studies in which we use queries to ask open-ended policy questions [Breaux and Antón 2005a], organize requirements into hierarchies [Breaux and Antón 2005a; Breaux et al. 2006] and identify some ambiguities and conflicts [Breaux et al. 2006].

(3) Provide a means to formalize and compare different stakeholder viewpoints. We support this goal in two ways: (1) by formally distinguishing between the words in a domain description and the engineer's interpretation of those words in a conceptual model; and (2) by providing formal semantics to express different stakeholder viewpoints on the same requirements model with regard to purposes [Breaux and Antón 2005b], transactions and delegations [Breaux et al. 2006].

In the remainder of this section, we present an introduction to Description Logic followed by the relevant terminology and formal definitions for Semantic Parameterization and the process to map domain descriptions into DL expressions.

3.1 Introduction to Description Logic

The Description Logic (DL) is a subset of first-order logic used to express and reason about knowledge [Baader et al. 2002]. In DL, knowledge is maintained in a knowledge base $\mathcal{KB}$ that is comprised of an intensional component, called the TBox $\mathcal{T}$, which describes abstract *terminology* or domain knowledge, and an extensional component, called the ABox $\mathcal{A}$, which describes *assertions* about a domain-specific problem. The TBox contains terminological axioms called *descriptions* that define both *concepts*, used to describe individuals, and *roles*, used to describe binary relationships between individuals. The ABox contains assertions about individuals in terms of concepts and roles. In the DL family $\mathcal{ALC}$ (Attributive Language + Complement), complex descriptions are built from other descriptions using constructors such as union, intersection, negation and full existential qualifiers over roles. In this paper, we use the DL family $\mathcal{ALCI}$ that combines the constructors from $\mathcal{ALC}$ with role inversion ($\mathcal{I}$) [Baader et al. 2002]. Reasoning in the DL family $\mathcal{ALCI}$ is known to be PSPACE-complete [Baader et al. 2002].

Consider a brief example in the health care domain. We define a TBox that contains descriptions for the concept `Hospital` and the role `hasPatient` and an ABox that contains assertions over two individuals x and y in the form `Hospital(`x`)` and `hasPatient(`x`,`y`)` with the following interpretations: `Hospital(`x`)` asserts that x belongs to the concept `Hospital` and `hasPatient(`x`,`y`)` asserts that the individual x belongs to the role `hasPatient` in which the individual y fills that role. By separating intensional knowledge (e.g., concepts and roles) from extensional knowledge (e.g, individuals), it is possible to make inferences about individuals using only the

concepts and roles they fill.

Reasoning in DL begins with an interpretation $\mathcal{I}$ that consists of a non-empty set $\Delta^{\mathcal{I}}$, called the domain of interpretation, and the interpretation function $\cdot^{\mathcal{I}}$ that maps concepts and roles to subsets of $\Delta^{\mathcal{I}}$ as follows: every atomic concept C is assigned a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every atomic role R is assigned the subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. In Description Logic, two special concepts are defined: $\top$, pronounced "top," with the interpretation $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot$, pronounced "bottom," with the interpretation $\bot^{\mathcal{I}} = \oslash$. In addition to constructors for union, intersection and negation, DL provides a constructor to constrain role values, written $R.C$, which means the filler for the role R belongs to the concept C. The interpretation function is extended to concept definitions for the DL family $\mathcal{ALCI}$ as follows, where C and D are concepts and R and S are roles in the TBox:

$$
\begin{aligned}
&\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \\
&\bot^{\mathcal{I}} = \oslash \\
&(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
&(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
&(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
&(\forall R.C)^{\mathcal{I}} = \left\{ a \in \Delta^{\mathcal{I}} \ \mid \ \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}} \right\} \\
&(\exists R.\top)^{\mathcal{I}} = \left\{ a \in \Delta^{\mathcal{I}} \ \mid \ \exists b.(a,b) \in R^{\mathcal{I}} \right\} \\
&(R^{-})^{\mathcal{I}} = \left\{ (b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \ \mid \ (a,b) \in R^{\mathcal{I}} \right\}
\end{aligned}
$$

Description Logic includes axioms for subsumption, equivalence and disjointness with respect to a TBox. Subsumption provides a means to describe individuals in terms of generalities and organize concepts into subsumption hierarchies, similar to class hierarchies in object-oriented design. We say a concept C is subsumed by a concept D, written $\mathcal{T} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the TBox $\mathcal{T}$. The concept C is equivalent to a concept D, written $\mathcal{T} \models C \equiv D$,

if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the TBox $\mathcal{T}$. The concept C is disjoint from D, written $\mathcal{T} \models C \sqcap D \rightarrow \bot$, if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \oslash$ for all interpretations $\mathcal{I}$ that satisfy the TBox $\mathcal{T}$.

For example, in the health care domain, we might encounter two phrases "to treat individuals" and "treatment of individuals" intended to have the same intensional meaning. In the first phrase, the word "treat" is an action verb whereas in the second phrase the word "treatment" is a noun that describes the activity "to treat someone." In DL, we formulate a definition using the following equivalence axiom: `Treatment` $\equiv$ `Activity` $\sqcap$ `hasAction.Treat`. The axiom states that the concept `Treatment` is equivalent to an `Activity` with the role `hasAction` whose filler is constrained to the concept `Treat`. This axiom ensures that descriptions expressed as either "to treat individuals" or "treatment of individuals" will be conceptually equivalent for reasoning purposes, regardless of the object of the treatment (e.g., individuals, patients, etc.)

3.2 Formal Definitions

In Semantic Parameterization, the universe of discourse is comprised of the concepts and roles contained in the TBox $\mathcal{T}$, assertions contained in the ABox $\mathcal{A}$, and the set of natural language words $\mathcal{W}$, called the *lexicon*, that consists of all words in the union of the following disjoint subsets: the set $\mathcal{N}$ of nouns, the set $\mathcal{J}$ of non-inflected adjectives and the set $\mathcal{V}$ of transitive and ditransitive verbs (excluding the verbs to-be and to-have); therefore, $\mathcal{W} = \mathcal{N} \cup \mathcal{J} \cup \mathcal{V}$. Adverbs that are derived from adjectives are mapped to their adjectival form in $\mathcal{J}$. The dictionary maps words in the lexicon to the concepts and roles in the TBox with statements about individuals expressed as assertions in the ABox. The following definitions precisely define the

dictionary as well as polysemy and synonymy that occur in domain descriptions:

Definition 3.1. The dictionary $\mathcal{D}$ is a subset of pairs in $\mathcal{W} \times \mathcal{T}$, called dictionary entries, that consist of a word $w \in \mathcal{W}$ and an axiom $A \in \mathcal{T}$. The axiom A is one of two possible DL descriptions: (1) a concept C; or (2) if $w \in \mathcal{N}$, the description can be a role $R.D$ for a some concept D. In the second case, the word w is used in natural language statements to refer to individuals that belong to the domain of the role description. For example, a dictionary entry (*subject*, `isSubjectOf.Activity`) contains the word *subject* that is used to refer to an individual x in the domain of the role `isSubjectOf(`x`,`y`)`, as opposed to the range of the role, which refers to an individual belonging to the concept `Activity`.

To improve readability, the concept and role names in dictionary entries correspond to their dictionary word as follows: for a word *word* in the dictionary, if the word refers to a concept we use the concept name `Word`, the exact word with capital initial, or if the word refers to a role we use the role name `isWordOf` with the inverse role name `hasWord` such that $\texttt{isWordOf}^{-} \equiv \texttt{hasWord}$.

Definition 3.2. Two words in a domain description are *synonyms* if they are different words and their uses have the same intensional or extensional meanings. Two different words w_1, w_2 are intensional synonyms, if for two dictionary entries $(w_1, A_1), (w_2, A_2) \in \mathcal{D}$, it is true that $\mathcal{T} \models A_1 \equiv A_2$; or they are extensional synonyms if the words are used to refer to the same set of individuals in the ABox such that, for all x in this set, $\mathcal{KB} \models A_1(x) \wedge A_2(x)$, recalling that if A_1 and A_2 are roles then the concerned individuals are in the domain of that role.

Definition 3.3. A word in a domain description is a *polyseme* if it has different

intensional or extensional meanings. The word w is an intensional polyseme, if for two dictionary entries $(w, A_1), (w, A_2) \in \mathcal{D}$, it is true that A_1 and A_2 are not equivalent or $(A_1)^{\mathcal{I}} \neq (A_2)^{\mathcal{I}}$ for some interpretation $\mathcal{I}$ that satisfies the TBox $\mathcal{T}$; or it is an extensional polyseme, if the word is used to refer to two different individuals x, y in the ABox such that $\mathcal{KB} \models A_1(x) \wedge A_2(y)$, recalling that if A_1 and A_2 are roles then the concerned individuals are in the domain of that role.

The Node-Edge Problem. The node-edge problem is the matter of deciding whether a word maps to a concept or a role in a conceptual model. For example, we can map the word *patient* to a concept `Patient` or to a role such as $\texttt{isPatientOf}_1\texttt{.Doctor}$ or $\texttt{isPatientOf}_2\texttt{.Hospital}$. The first role may describe a patient who has been assigned to a doctor whereas the second role may describe a patient who has been admitted to a hospital. To resolve this problem, we assert that for a common word w and a set of conceptually related roles $\{ R_i \mid (w, R_i) \in \mathcal{D}$ for $1 \leq i \leq n \}$, there exists a shared concept C with dictionary entry (w, C) and the subsumption axiom $R_1 \sqcup R_2 \sqcup ... \sqcup R_n \sqsubseteq C$ in the TBox. For example, the axiom $\texttt{isPatientOf}_1 \sqcup \texttt{isPatientOf}_2 \sqsubseteq \texttt{Patient}$ ensures that individuals who belong to either of the roles $\texttt{isPatientOf}_1$ or $\texttt{isPatientOf}_2$ also belong to the concept `Patient`. The advantage of using a role to refer to an individual is increased specificity (e.g., it implies a relation to another concept in the range of that role) whereas using a concept to refer to an individual provides the freedom to generalize among similar individuals irrespective of their different relations to other concepts.

3.3 Parameterization Process

The parameterization process extends the ABox with assertions acquired from a domain description and allows the engineer to develop re-usable NL patterns that generalize across multiple descriptions. The extension is partitioned into two sets of DL assertions: (1) those assertions that come from words in the NL statement, called the *grounding*; and (2) those assertions that come from words inferred by the engineer, called the *meta-model*. Each new assertion incrementally builds a specification; a notion Zave and Jackson call conjunction as composition [Zave and Jackson 1993]. Together, the grounding and meta-model align with the NL phrase structure to comprise the NL pattern. Engineers who re-use these patterns will improve consistency in requirements models because the resulting models are structurally similar under DL subsumption and unification. In addition, these engineers will save time and effort because the NL patterns serve as templates that characterize the tacit knowledge in conceptually related domain descriptions.

We illustrate the parameterization process with the Unstructured Natural Language Statement (UNLS) $\text{UNLS}_{1.0}$, grounding $\mathcal{G}$ and meta-model $\mathcal{M}$ in Table I. We assume the dictionary contains all the necessary words for this exercise with corresponding concepts and roles contained in the TBox. In practice, however, the engineer may need to add new words to the dictionary which may require adding new DL axioms for subsumption, equivalence and disjointness. The process proceeds in three phases: (1) apply phrase heuristics to disambiguate words with an anaphoric or cataphoric function such as pronouns to identify extensional synonyms; (2) derive the grounding by using the dictionary to assign DL meanings to

Table I. The grounding $\mathcal{G}$ and meta-model $\mathcal{M}$ derived from $\text{UNLS}_{1.0}$

Domain Description	
UNLS$_{1.0}$: The customer$_{1,1}$ must not share$_{2,2}$ the access-code$_{3,3}$ of the customer$_{1,1}$ with someone$_{4,4}$ who is not the provider$_{5,4}$.	
Grounding ($\mathcal{G}$)	**Meta-Model ($\mathcal{M}$)**
`Customer`(p_1) ⊓ `Share`(p_2) ⊓ `isAccessCodeOf`(p_3,p_1) ⊓ `Someone`(p_4) ⊓ `Provider`(p_4)	`Activity`(p_5) ⊓ `hasSubject`(p_5,p_1) ⊓ `hasAction`(p_5,p_2) ⊓ `hasObject`(p_5,p_3) ⊓ `hasTarget`(p_5,p_4) ⊓ `isRefrainmentOf` (p_5,p_1)

the words in the domain description; (3) derive the meta-model by using the dictionary to identify the tacit relationships between concepts implied by the domain description. We prepared $\text{UNLS}_{1.0}$ by hyphenating nouns in the statement that describe a single concept or role. For example, the noun "access code" is hyphenated because it refers to a single concept `AccessCode` or role `isAccessCodeOf`.

In the first phase, the engineer coordinates with domain experts and applies phrase heuristics to disambiguate references between different noun phrases that refer to the same person, place or thing called an *anaphoric* or *cataphoric* function. Pronouns (e.g., he, her, it, this, that, etc.) and nouns that follow articles (e.g., a, the) both refer to a unique person, place or thing in domain descriptions. Because pronouns are frequent sources of ambiguity, the engineer must identify and replace pronouns with a definite article followed by the noun phrase that uniquely identifies the intended individuals (e.g., replace *it* with *the system* if *it* refers to *the system.*). For the same reason, possessive pronouns are replaced by the engineer (e.g., *their website* is replaced with *the company's website* if the possessive pronoun *their* refers to *the company*).

The engineer must then identify and distinguish intensional and extensional syn-

onyms and polysemes. For example, if the same noun is used to refer to two different individuals (e.g., *this network* and *that network*) then the engineer must consistently distinguish these extensional polysemes. We assume all lexically equivalent words are intensional synonyms (same concept) and extensional polysemes (different individuals), unless otherwise distinguished using subscripts as follows: for $network_{x,y}$, the subscript x is an intensional index and the subscript y is an extensional index. Similar indices are synonyms and dissimilar indices are polysemes for the given word *network*. For example, the words $network_{1,1}$ and $network_{1,2}$ represent two intensional synonyms and extensional polysemes (e.g., same concept but different individuals).

In $\text{UNLS}_{1.0}$, for example, the two occurrences of the word *customer* both have the same intensional and extensional meaning, whereas the word *someone*, which describes any person, and the word *provider*, which in this context describes a person who provides services, both have different intensional meanings (different concepts) but have the same extensional meaning (same individual).

In the second phase, the engineer then builds an extension $\mathcal{G}$ to the ABox $\mathcal{A}$. For each word $w_{x,y}$ in $\text{UNLS}_{1.0}$, find the dictionary entry $(w,\ A) \in \mathcal{D}$ and extend the ABox with assertions $C(p_y)$ for individual p_y, if A is a concept C, or $R(p_y, p_v)$ for individual p_y, p_v, if A is a role R, noting that the individual p_v may be the same individual from another word $w_{u,v}$ in the prepared statement. For example, in $\text{UNLS}_{1.0}$, since the phrase "access-code$_{3.3}$ of the customer$_{1.1}$" denotes an association between the *customer* and the *access-code*, we add the assertion `isAccessCodeOf(`p_3`,`p_1`)` where individual p_3 refers to the *access-code* and p_1 refers to

Table II. Primitive RNLS Heuristics

Heuristic	RNLS and ABox Extension
(i)	RNLS: The $word_C$ is a $word_D$. For some $(word_C, C), (word_D, D) \in \mathcal{D}$, find an individual x such that: $\mathcal{A}'' = \mathcal{A}' \cup \mathcal{A} \cup \{C(x) \wedge D(x)\}$
(ii)	RNLS: The $word_R$ of a $word_C$. For some $(word_C, C), (word_R, R.C) \in \mathcal{D}$, find individuals x, y such that: $\mathcal{A}'' = \mathcal{A}' \cup \mathcal{A} \cup \{R(x,y) \wedge C(y)\}$
(iii)	RNLS: The $word_C$ has a $word_R$. For some $(word_C, C), (word_R, R.C) \in \mathcal{D}$ find individuals x, y such that: $\mathcal{A}'' = \mathcal{A}' \cup \mathcal{A} \cup \{C(x) \wedge R^{-}(x,y)\}$
(iv)	RNLS: The $word_D$ is a $word_R$ of a $word_C$. For some $(word_D, D), (word_R, R.C), (word_C, C) \in \mathcal{D}$ find individuals x, y such that: $\mathcal{A}'' = \mathcal{A}' \cup \mathcal{A} \cup \{D(x) \wedge R(x,y) \wedge C(y)\}$

the *customer*. However, if the phrase were simply "access-code$_{3.3}$" with no mention of customer, we would have added the assertion `AccessCode(`p_3`)`. After completing phase one, the extended ABox $\mathcal{A}'$ includes the grounding $\mathcal{G}$ from Table I as follows:

$\mathcal{A}' = \mathcal{A} \cup \mathcal{G}$

At this point, all of the nouns, verbs and adjectives in UNLS$_{1.0}$ have been formalized using DL; these words and derived assertions in the above extension are called the *grounding*. In the third phase, the engineer elicits from domain experts the implicit or tacit knowledge, if any, that relates the individuals in the grounding to each other through a sequence of implied roles. Table II provides a set of heuristics based on primitive restricted natural language statements (RNLS) that only use the verbs to-be and to-have. Applying the primitive RNLS heuristics will introduce new words and assertions that comprise the *meta-model*.

In Table II, $word_C$, $word_D$, and $word_R$ are words in the dictionary $\mathcal{D}$; C, D are concepts and R is a role in the TBox $\mathcal{T}$; and x and y are individuals. The articles *the*, *a*, *an* in the primitive RNLS may be interchanged as necessary, since the uniqueness expressed by these words maps to extensional references and individuals in the UNLS and DL formulas, respectively. In addition, the engineer may find that the

Table III. Results from Applying Phrase Heuristics

Index	Heuristic	Resulting RNLS
(1a)	(ii)	The **access-code**$_{3,3}$ of the **customer**$_{1,1}$.
(1b)	(iv)	The **access-code**$_{3,3}$ is the property$_{6,3}$ of the **customer**$_{1,1}$.
(1c)	(iv)	The **access-code**$_{3,3}$ is a possession$_{7,3}$ of the **customer**$_{1,1}$.
(2)	(iv)	The **customer**$_{1,1}$ is the subject$_{8,1}$ of an activity$_{9,5}$.
(3)	(iv)	**Share**$_{2,2}$ is the action$_{10,2}$ of an activity$_{9,5}$.
(4)	(iv)	The **access-code**$_{3,3}$ is the object$_{11,3}$ of an activity$_{9,5}$.
(5)	(iv)	**Someone**$_{4.4}$ is the target$_{12,4}$ of an activity$_{9,5}$.
(6)	(iii)	The activity$_{9,5}$ has a subject$_{8,1}$, action$_{10,2}$, object$_{11,3}$ and target$_{12,4}$.
(7)	(iv)	The activity$_{9,5}$ is a refrainment$_{13,5}$ of the **customer**$_{1.1}$.

application of heuristic (ii) or (iii) can be restated as heuristic (iv) by substituting the $word_R$ used in heuristic (ii) or (iii) with $word_D$ in heuristic (iv) and finding a new word $word_R$ that satisfies heuristic (iv). Table III shows the results of applying these primitive RNLS heuristics from Table II to UNLS$_{1.0}$. Each row in Table III consists of: a numbered index to be used in the following discussion; the roman numeral index of the heuristic applied from Table II; and the RNLS that results from applying this heuristic. In each result, all of the grounding words are boldface.

Beginning with the phrase "access-code of the customer," applying the primitive RNLS heuristic (ii) yields the RNLS (1a) in Table III. However, using the primitive RNLS heuristic (iv) by finding a new word for $word_D$, the engineer may elicit RNLS (1b) and (1c), exploring the relationship of the access code to the customer as either the property or possession of the customer. The distinction may have legal consequences, because in certain jurisdictions it may be illegal for a provider to revoke an access code from their customer when the access code is owned by the customer (e.g., the code is the customer's property). For this reason, the engineer must ensure the meta-model describes the viewpoint of the appropriate stakeholder(s) so that the model is consistent with the intended interpretation of the overall environment. For the purpose of this illustration, we choose RNLS (1a).

RNLSs (2)-(5) are elicited by recognizing that the $\text{UNLS}_{1.0}$ describes an activity, in which the customer must not share their access code. This implied activity contributes a new individual p_5 to the meta-model. Each word (customer, share, access-code, and someone) relevant to this activity is assigned a role (subject, action, object, and target) in the activity.

Activities may be composed differently by different engineers. For example, in Table III the word "subject" could be substituted for the word "actor" in RNLS (2) and RNLS (6). Likewise, one might designate the subject and object as roles of an action, not an activity. Because the dictionary ensures that words in the UNLS are individually mapped to concepts and roles, such variations can be aligned using the equivalence and subsumption axioms in the TBox. Finally, the modal phrase "must not" in $\text{UNLS}_{1.0}$ designates the activity as something the customer should not do, which we call a refrainment in RNLS (7). We conclude the parameterization process by adding the new assertions that comprise the meta-model $\mathcal{M}$ to the extended ABox as follows: $\mathcal{A}'' = \mathcal{A} \cup \mathcal{G} \cup \mathcal{M}$

Recall that one goal in the parameterization process is to relate individuals through a sequence of roles to identify the tacit relationships between individuals. In this example, that sequence of roles is: `isAccessCodeOf(`p_3`,`p_1`), hasSubject(`p_5`,`p_1`), hasAction(`p_5`,`p_2`), hasObject(`p_5`,`p_3`), hasTarget(`p_5`,`p_4`), isRefrainmentOf(`p_5`,`p_1`).` In this application of the parameterization process, the phrase structure of the domain description generally describes an actor who performs an action on an object. We generalize this phrase structure into the basic activity pattern that appears in Section 4.

Like other forms of conceptual modeling, including object-oriented design, the meta-modeling process requires engineers to investigate and expose the tacit or implicit relationships implied by domain descriptions. Because this process is intensive, engineers should first try to re-use previously identified RNLS patterns, such as those presented in Section 4, and reserve the meta-modeling process for modeling domain descriptions that contain new phrase structures. Likewise, the failure to apply an existing pattern may be due to variations in the phrase structures that yield either one or more new patterns, extensions to existing patterns, or ambiguities with existing patterns. With regard to ambiguities, consider the following statement: "The health plan notifies patients with e-mail." The phrase "with e-mail" has two different meanings expressed in DL and contained in the meta-model: (1) the patient has e-mail, meaning the patient can access an e-mail account; or (2) the health plan uses e-mail to notify the patient. In Semantic Parameterization, the first meaning is expressed as `Patient` $\sqcap$ `hasEmail` whereas the second meaning is expressed as `Activity` $\sqcap$ `hasInstrument.Email` (the pattern for the second meaning is detailed in Section 4.2.4). By documenting NL patterns that contain the same or similar phrase structure, engineers can identify these ambiguities so that domain experts can decide which meaning is intended by the description.

4. RESTRICTED NATURAL LANGUAGE STATEMENTS

The Semantic Parameterization process yields re-usable natural language patterns that are realized as simple sentences called Restricted Natural Language Statements (RNLS). Based on our experience working within two different domains, health care and finance, we found that most domain descriptions can be partially mapped into

a formal model using at least one of the RNLS patterns presented in this section.

The RNLS patterns are organized into the following three categories:

(1) **Primitive RNLS** that use the verbs to-be and to-have (see Table II);

(2) **Basic and extended activities**, including transactions; and

(3) **References to other activities**, including:

  (a) verb phrases masquerading as nouns and adjectives;

  (b) transitive verbs followed by verb phrases;

  (c) nouns distinguished by verb phrases; and

  (d) purposes and instruments.

The RNLS patterns share a common meta-model that use one or more roles listed in Table IV. These roles map assertions in the ABox to questions in the Inquiry Cycle Model (ICM) [Potts et al. 1994]. For each role, the answers to these questions are other concepts and roles to which the filler of that role belongs.

Table IV. Mapping from DL roles to questions in the Inquiry Cycle Model

DL Role in Meta-model	**ICM Question**
`isSubjectOf.Activity`	*Who* performs the action?
`isObjectOf.Activity`	Upon *what* is the action performed?
`isTargetOf.Activity`	With *whom* is the transaction performed?
`isPurposeOf.Activty`	*Why* is the action performed?
`isInstrumentOf.Activity`	*How* is the action performed?
`isLocationOf.Activity`	*Where* is the action performed?

In Semantic Parameterization, each natural language pattern is comprised of: a domain description expressed as an RNLS; the DL expression that maps to the RNLS; and the DL pattern that generalizes the DL expression for re-use. In the DL pattern, we change domain-specific concepts that appear in the grounding to the generalized concept `Noun` or `Verb` to indicate which dictionary entries can be used

Table V. Basic activity pattern with subject, action and object

Natural Language Statements	
RNLS$_2$: The provider promptly updates erroneous information.	
Expression	**Pattern**

`Activity ⊓ Prompt`
`⊓ hasSubject.Provider`
`⊓ hasAction.Update`
`⊓ hasObject.(Information`
`⊓ Erroneous)`
	`Activity`
	`⊓ hasSubject.Noun`
	`⊓ hasAction.Verb`
	`⊓ hasObject.Noun`

to map concepts to these slots. For example, the DL expression `hasAction.Verb` indicates that only concepts whose dictionary word is in the set of verbs $\mathcal{V}$ may constrain the role `hasAction`.

4.1 Basic and Extended Activities

RNLS with verbs other than the irregular verbs to-be and to-have share a common pattern called the *activity pattern*. This pattern consists of four dictionary words: the word *activity* which defines a concept with three properties, including the *subject* (a noun) who performs an *action* (a verb) on some *object* (a noun or verb phrase). Adverbs that modify a verb (the action) are changed to adjectives that describe the activity (e.g., "to confidentially share" refers to an activity that is "confidential" with an action "share").

Table V shows an example of the activity pattern in which the concepts for the dictionary words *provider*, *update* and *information* constrain the range of the roles `hasSubject`, `hasAction`, and `hasObject`, respectively. The adverb "promptly" is changed to the non-inflected adjective *prompt* that describes the activity in the intersection (`Activity ⊓ Prompt`). Likewise, the adjective "erroneous" describes the information in the intersection (`Information ⊓ Erroneous`).

Transactions are an extension of the basic activity pattern and are identified

by special verbs, including disclose, share, send, rent, etc. Transactions require a fourth role `hasTarget` that designates a supplemental actor who participates in the action with the subject. For example, the phrase "to send electronic mail" has the action "send" that requires a *target* to whom the object (electronic mail) is sent. We model this *target* property of transactions using the role `hasTarget.Noun`.

4.2 References to other activities

In domain descriptions, a single natural language statement may describe relationships between multiple activities. For example, the phrase "share information for marketing" refers to two activities in which the second activity "marketing" is the *purpose* of performing the first activity "share information." The engineer applies RNLS patterns to separate these activities into distinct RNLS that are linked using nested references to maintain the intended meaning. We discuss the following four RNLS patterns in detail: verb phrases masquerading as nouns and adjectives; transitive verbs followed by verb phrases; distinguishing nouns by verb phrases; and purposes and instruments.

4.2.1 *Verb phrases masquerading as nouns and adjectives.* Nouns that end in -ing (called gerunds) and other nouns that end in -ance, -sion, -tion, -ism, -sure, -zure, and -ment often describe activities that may be expanded into verb phrases and separate RNLS. These nouns may follow transitive verbs (see Section 4.2.2) or appear as the purpose or instrument (see Section 4.2.4). These nouns often are lexically similar to the verb in the expanded verb phrase, for example: permission/ permit, restriction/ restrict, requirement/ require, etc. During restatement, the engineer is required to: (1) replace the noun with a cross-reference to a separate

Table VI. Verb phrases masquerading as nouns

Natural Language Statements	Expression

UNLS$_{3.0}$: The provider documents disclosures of patient information.
RNLS$_{3.1}$: The provider documents (**RNLS**$_{3.2}$).
RNLS$_{3.2}$: Someone discloses patient information to someone.
	`Activity`
	`  ⊓ hasSubject.Provider`
	`  ⊓ hasAction.Document`
	`  ⊓ hasObject.(Disclosure`
	`  ⊓ hasSubject.Someone`
	`  ⊓ hasAction.Disclose`
	`  ⊓ hasObject.PatientInformation`
	`  ⊓ hasTarget.Someone`
	`  )`

RNLS that will become the expanded verb phrase; (2) set the verb tense in the expanded verb phrase to present-simple tense; and (3) state the explicit or implicit subject or object of the verb phrase in the new RNLS.

In UNLS$_{3.0}$ presented in Table VI, the noun "disclosure" is expanded from RNLS$_{3.1}$ to RNLS$_{3.2}$ by using the ambiguous noun "someone" to conservatively describe an extensional polyseme. In the TBox, we ensure the following axiom is defined to complement this pattern: $\mathcal{T} \models$ `Disclosure` $\equiv$ `Activity` $\sqcap$ `hasAction.Disclose`. We frequently encountered this type of intensional knowledge in the HIPAA case study [Breaux et al. 2006] discussed later in Section 5.

Adjectives that are derivable from past-tense verbs describe activities and can be expanded into separate RNLS. These adjectives are sometimes called transitive adjectives [Keenan and Faltz 1985]. For example, the phrase "the disclosed information" has the adjective "disclosed" that is also a past-tense verb. The noun that follows these adjectives is always the object of the described action. Thus, we axiomatize this relationship in the TBox as follows: $\mathcal{T} \models$ `Disclosed` $\equiv$ `isObjectOf.(Activity` $\sqcap$ `hasAction.Disclose)`.

Table VII. Transitive verbs followed by verb phrases

Natural Language Statements	
UNLS$_{4.0}$:	The provider restricts sharing information with third-parties.
RNLS$_{4.1}$:	The provider$_{1.x}$ restricts (**RNLS**$_{3.2}$).
RNLS$_{4.2}$:	The provider$_{1.x}$ shares information with a third-party.

Expression	Pattern

`Activity`
`  ⊓ hasSubject.Provider`
`  ⊓ hasAction.Restrict`
`  ⊓ hasObject.(Activity`
`  ⊓ hasSubject.Provider`
`  ⊓ hasAction.Share`
`  ⊓ hasObject.Information`
`  ⊓ hasTarget.ThirdParty`
`  )`
	`Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.(Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  )`

4.2.2 *Transitive verbs followed by verb phrases.* Transitive verbs in domain descriptions such as restrict, limit, allow, deny, notify, require and recommend may be followed by verb phrases. During restatement, the engineer is required to: (1) replace the verb phrase with a cross-reference to a separate RNLS that will contain the verb phrase; (2) change the verb in the verb phrase from present-continuous to present-simple tense; and (3) state the explicit or implicit subject, object and target of the verb phrase in the new RNLS, if any. For unstated (implicit) subjects, one may use the same subject from the unrestricted statement if that assumption is correct or elicit the correct subject from the domain expert.

In UNLS$_{4.0}$ in Table VII, the transitive verb "restrict" is followed by the verb phrase "sharing information with third-parties." Therefore, we separate the verb phrase from RNLS$_{4.1}$ into RNLS$_{4.2}$ and cross-reference, accordingly. The subject of the verb phrase is unspecified, so we assume the explicit subject from RNLS$_{4.1}$ (the provider) in which RNLS$_{4.2}$ is nested will suffice; this assumption may not always be valid and any final decisions should be checked with relevant stakeholders. To derive the DL formula, we apply the activity pattern to RNLS$_{4.2}$ and assign the

Table VIII. Nouns distinguished by verb phrases

Natural Language Statements	
UNLS$_{5.0}$:	The provider notifies customers who receive health services.
RNLS$_{5.1}$:	The provider notifies the customer$_{1.x}$ who (**RNLS**$_{5.2}$).
RNLS$_{5.2}$:	The customer$_{1.x}$ receives health services.

Expression	Pattern

`Activity`
`  ⊓ hasSubject.Provider`
`  ⊓ hasAction.Notify`
`  ⊓ hasTarget.(Customer`
`  ⊓ isSubjectOf.(Activity`
`  ⊓ hasAction.Receive`
`  ⊓ hasObject.HealthService`
`  )`
`  )`
	`Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasTarget.(Noun`
	`  ⊓ isSubjectOf.(Activity`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  )`
	`  )`

resulting DL expression to the role `hasObject` in the formula derived from RNLS$_{4.1}$.

4.2.3 *Nouns distinguished by verb phrases.* Verb phrases also serve as constraints that distinguish nouns in domain descriptions. For nouns that signify a person, place or thing, the words "who," "where" and "that," respectively, will often precede these verb phrases in domain descriptions. The engineer must separate the verb phrase(s) into new RNLS, replacing the original verb phrase with a cross-reference to the new RNLS, and change the verb tense in the new RNLS to present-simple.

In UNLS$_{5.0}$ presented in Table VIII, the act to notify does not apply to all customers; rather it is limited to only those customers "who receive health services." The word *customer* in RNLS$_{5.1}$ and RNLS$_{5.2}$ is an extensional synonym that preserves the meaning from UNLS$_{5.0}$ after the separation. In this example, the object of the notification is missing, thus the engineer must elicit this information from domain experts to disambiguate the statement.

To derive the DL formulas, we apply the activity pattern to RNLS$_{5.2}$ and retopicalize the derived formula for the customer by: inverting the role `hasSubject.Customer`

Table IX. Activities with a purpose (The Why)

Natural Language Statements	
UNLS$_{6.0}$:	The provider discloses information to market services to individuals.
RNLS$_{6.1}$:	The provider discloses information to (**RNLS**$_{6.2}$).
RNLS$_{6.2}$:	Someone markets services to individuals.

Expression	Pattern

`Activity`
`  ⊓ hasSubject.Provider`
`  ⊓ hasAction.Disclose`
`  ⊓ hasObject.Information`
`  ⊓ hasPurpose.(Activity`
`  ⊓ hasAction.Market`
`  ⊓ hasObject.Service`
`  ⊓ hasTarget.Individual`
`  )`
	`Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  ⊓ hasPurpose.(Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  )`

to the intersection `(Customer ⊓ isSubjectOf)` with the remaining description of that activity `(Activity ⊓ hasAction ⊓ hasObject)` assigned to be the constraint on the role `isSubjectOf`. As shown in Table VIII, the re-topicalized DL formula from RNLS$_{5.2}$ constrains the role `hasTarget` in the DL formula derived from RNLS$_{5.1}$.

4.2.4 *Purposes and instruments: the why and the how.* The purpose (also called cause or justification) answers the question "why an action is performed" whereas the instrument (also called the strategy or method) answers the question "how an action is performed." These two properties also appear in goal hierarchies from requirements engineering [van Lamsweerde 2000], in which higher goals (purposes) are refined into lower goals (instruments). The purpose and instrument appear in domain descriptions as either: (1) a verb phrase; or (2) a noun masquerading as a verb phrase. In the first case, we apply the activity pattern, whereas, in the second case, we apply the pattern discussed in Section 4.2.1.

In UNLS$_{6.0}$ in Table IX, the purpose "to market services to individuals" is a verb phrase that answers why the "provider discloses information." To apply this pattern, the engineer separates the verb phrase into RNLS$_{6.2}$, but this time they

Table X. Activities with an instrument (The How)

Natural Language Statements	
UNLS$_{7.0}$:	The employee protects documents by encrypting them.
RNLS$_{7.1}$:	The employee$_{1.x}$ protects a document$_{1.y}$ by (**RNLS**$_{7.2}$).
RNLS$_{7.2}$:	The employee$_{1.x}$ encrypts the document$_{1.y}$.

Expression	Pattern

`Activity`
`  ⊓ hasSubject.Employee)`
`  ⊓ hasAction.Protect`
`  ⊓ hasObject.Document`
`  ⊓ hasInstrument.(Activity`
`  ⊓ hasSubject.Employee`
`  ⊓ hasAction.Encrypt`
`  ⊓ hasObject.Document`
`  )`
	`Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  ⊓ hasInstrument.(Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  )`

make no assumptions about the implicit subject. The formula derived from RNLS$_{6.2}$ is assigned to be a constraint on the role `hasPurpose` in the formula derived from RNLS$_{6.1}$.

In Table X, the UNLS$_{7.0}$ illustrates the instrumental phrase "by encrypting them," that answers how the employee "must protect documents." To apply this pattern, the engineer separates the instrumental phrase into a separate RNLS$_{7.2}$. The DL expression derived from RNLS$_{7.2}$ is assigned to be a constraint on the role `hasInstrument` in the expression for RNLS$_{7.1}$.

In some cases, the instrumental phrase may contain the verb "use", such as "by using AES" or "by using encryption" where "AES," which stands for Advanced Encryption Standard, is an encryption algorithm[1] and "encryption" is the activity "to encrypt." In the first case, if the noun that follows "using" is a thing, not an activity, then the engineer should apply the activity pattern to the entire verb phrase. However, in the second case, the engineer may wish to remove the superfluous verb "using" and parameterize the noun using the pattern for verb phrases

[1]National Institute of Technology, FIPS Pub. 197

masquerading as nouns in Section 4.2.1.

5. VALIDATION AND PROCESS EVOLUTION

Semantic Parameterization and the RNLS patterns have been developed using Grounded Theory [Glaser and Strauss 1967] and validated in the following three studies. In each study, limitations in the parameterization process and formal models were identified and addressed before conducting subsequent studies. The limitations were either resolved by extending the NL patterns or they required new logical operators beyond the scope of Description Logic, such as arithmetic, deontic and temporal operators. Finally, every statement in the domain description was parameterized to avoid overlooking limitations in our process. The three studies are identified as follows:

(1) **Goals:** A formative study using the 100 most frequent, semi-structured goals from over 1200 goals acquired by applying GBRAM to over 100 Internet privacy policies in the finance and health care domains [Breaux and Antón 2005b; 2005a].

(2) **Facts:** A pilot study using the fact sheet text [DHHS 2003a] summarizing the U.S. HIPAA Privacy Rule for patients in which we extracted 19 business rules [Breaux and Antón 2005c].

(3) **Rules:** A case study using the regulatory text of the HIPAA Privacy Rule, four sections §164.520-526 [DHHS 2003b] in which we extracted 46 rights and 80 obligations governing access, consent, notification, and review of privacy practices [Breaux et al. 2006].

Table XI. Comparative overview of the three studies

Feature Description	**Goals**	**Facts**	**Rules**
Number of words in the domain description	868	1,700	5,900
Number of words in the grounding	708	318	1587
Number of words in the meta-model	905	341	1603
Number of formal models acquired	101	30	94
Number of dictionary entries used	187	140	234
Number of person hours spent during RNLS restatement	1	11	14
Number of person hours spent during parameterization	7	3	10
Percentage Domain Knowledge	45.5%	48.3%	49.7%

All of the RNLS patterns presented in Section 4 were acquired during the Goals study, except for the pattern to identify verb phrases masquerading as nouns, which was identified in the Rules study. The Facts and Rules studies served to test whether Semantic Parameterization and the RNLS patterns would scale from semi-structured goal descriptions to unstructured natural language documents, in particular, the legal language of U.S. government health care regulations.

In Table XI, we present measures to compare the three studies, including the number of words that appear in the domain descriptions, the grounding and the meta-model; the number of formal models and dictionary entries acquired; the number of hours spent applying the process; and the percent of domain-dependent knowledge in each study. In the Goals study, one goal description was found to describe two goals, resulting in the 101st model acquired during that study. The separation of activities into RNLS resulted in this finding. Because the semi-structured goal descriptions acquired using GBRAM have a similar phraseology to the RNLS patterns, we observe a higher ratio of grounding words to description words at 82% in the Goals study compared to 19% and 27% in the Facts and Rules studies, respectively. The Facts and Rules studies were conducted using unstructured natural language descriptions which contain articles, coordinators (and, or) and sub-ordinators (if, unless, except) that map to meta-model words and not ground-

Table XII. Usage for RNLS patterns in three studies

RNLS Pattern Name (Section)	**Goals**	**Facts**	**Rules**
Basic activity pattern (4.1)	280	132	613
Verb phrases masquerading as nouns and adjectives (4.2.1)	119	54	164
Transitive verbs (4.2.2)	18	14	75
Nouns distinguished by verb phrases (4.2.3)	20	5	22
Purposes (4.2.4)	13	23	45
Instruments (4.2.4)	20	3	26

ing words. The fewer verbs in individual goal statements also accounts for the fewer hours spent during RNLS restatement compared to the Facts and Rules studies. The number of case splits corresponds to the number of separate goals generated from logical disjunctions in a domain description. For example, in the goal description "providers and third-parties must notify patients of their privacy practices," the obligations of the providers and third-parties are interpreted as independent. Therefore, the English conjunction "and" is mapped to a logical disjunction which yields two separate goals, one for providers and the other for third-parties; this separation is called a case split. For the number of grounding words g and the number of meta-model words m, the percentage of domain knowledge is calculated by the simple formula: $d = g \ / \ (g + m)$. The percentage of meta-model reuse is $1 - d$, which is above 50% in each of these three studies. This percentage shows that the six roles presented in Table IV that comprise the meta-model frequently recur and account for a significant portion of the formalized domain descriptions.

In Table XII, we present the total number of occurrences in the three studies of the RNLS patterns from Section 4. The pattern name and the section from this paper in which it is discussed appears in the first column; the number of times each pattern was applied for each of the three studies described above appear in subsequent columns. The total number of occurrences for the basic activity pattern

include those activities acquired from applying all of the RNLS patterns described in Section 4.

The validation to date has addressed the reuse of the RNLS patterns to derive models from unstructured natural language domain descriptions. Limitations include arithmetic operators and temporal constraints for which Description Logic is unsuitable. Additional validation is needed to ascertain the ease with which engineers can apply these patterns compared to other conceptual modeling formalisms. We have developed tool support including a context-free grammar that has a formal semantics in Description Logic and a parser to read the semantic models and perform queries; these tools were used to obtain the results in Table XI and XII. We are currently comparing our approach to another called Ontological Semantics [Nirenburg and Raskin 2004] that requires reconciling domain descriptions with an upper ontology that spans several domains.

6. DISCUSSION AND SUMMARY

The formalization of requirements in Description Logic using Semantic Parameterization supports the Inquiry Cycle Model (ICM) in two ways: (1) it enables automating open-ended queries over formalized requirements [Breaux and Antón 2005a]; and (2) it organizes requirements into specialization hierarchies using the subsumption inference of Description Logic [Breaux et al. 2006]. In addition to subsumption inference, these techniques build upon the formalization of the ICM questions in Table IV and the RNLS patterns previously presented in Section 4.

Table XIII shows the results of a query that was applied to the 100 parameterized goals in the Goals study that was previously introduced in the Section 5

Table XIII. Answering two ICM questions from Table IV: *"what information can be shared?"* and *"with whom?"*

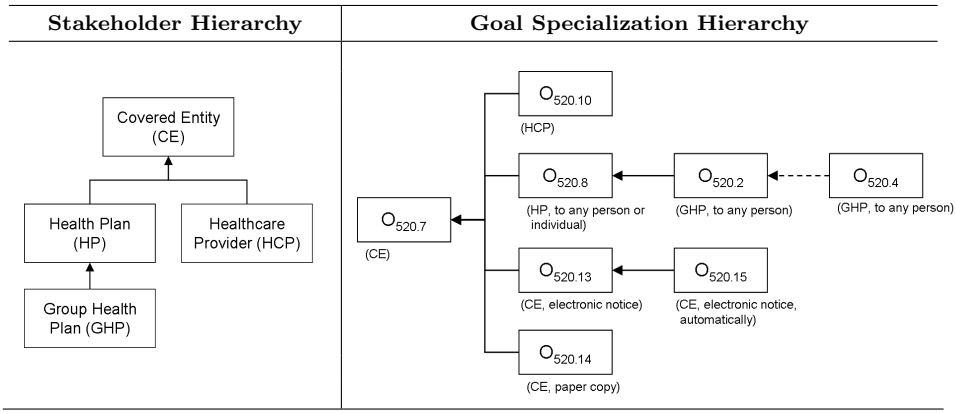
Goal ID	isObjectOf.Activity	isTargetOf.Activity
155	TransactionInformation	Subsidiary
155	ExperienceInformation	Subsidiary
822	Personally Identifiable Information (PII)	Affiliate
822	PII	ServiceProvider
954	Information	ThirdParty
954	Statistics	ThirdParty
156	TransactionInformation	Affiliate
156	ExperienceInformation	Affiliate
170	PII	Subsidiary

[Breaux and Antón 2005a]. The following axioms were first defined in the TBox $\mathcal{T} \models (\texttt{TransactionInformation} \sqcup \texttt{ExperienceInformation} \sqcup \texttt{PII} \sqcup \texttt{Statistics}) \sqsubseteq \texttt{Information}$. The question *"what information can be shared and with whom?"* was mapped to the two DL queries $\texttt{Information} \sqcap \texttt{isObjectOf.(Activity} \sqcap \texttt{hasAction.Share} \sqcap \texttt{hasTarget.}\top)$ and $\texttt{isTargetOf.(Activity} \sqcap \texttt{hasAction.Share} \sqcap \texttt{hasObject.Information})$ and unified with the knowledge base $\mathcal{KB}$ to obtain the results in Table XIII. The first column lists the Goal ID, where duplicate IDs indicate multiple answers caused by the case splitting of logical disjunctions. Recall that Table IV shows the DL formalization of the ICM questions as DL roles. In Table XIII, the individuals that belong to the roles `isObjectOf.Activity` and `isTargetOf.Activity` answer the question for *"what information is shared"* and *"with whom,"* respectively.

Table XIV shows a set of goal descriptions that all pertain to provision of notice and that were parameterized in the Rules study [Breaux et al. 2006]. The domain description defines a stakeholder specialization hierarchy that includes the covered entity (CE), the health plan (HP), the group health plan (GHP) and the health-care provider (HCP). The stakeholder hierarchy is realized in the following axioms: $\mathcal{T} \models \texttt{GHP} \sqsubseteq \texttt{HP}$ and $\mathcal{T} \models (\texttt{HP} \sqcup \texttt{HCP}) \sqsubseteq \texttt{CE}$. Using the stakeholder hierarchy, the goals are organized into a goal specialization hierarchy (displayed horizontally) that compares

Table XIV. Organizing Goals by Inferring Specialization Hierarchies

Eight Goal Descriptions Pertaining to Provision of Notice	
$O_{520.2}$:	The GHP must provide notice to any person.
$O_{520.4}$:	The GHP is not required to provide notice to any person.
$O_{520.7}$:	The CE must provide notice to any person or individual.
$O_{520.8}$:	The HP must provide notice to any person or individual.
$O_{520.10}$:	The HCP must provide notice to the individual.
$O_{520.13}$:	The CE must provide electronic notice to the individual.
$O_{520.14}$:	The CE must provide a paper copy of the notice to the individual.
$O_{520.15}$:	The CE must automatically provide electronic notice to the individual.



the goals by the roles `isSubjectOf`, `isActionOf`, `isObjectOf` and `isTargetOf` in the goal description as well as by adverbs and adjectives that modify verbs and nouns, respectively. In the goal hierarchy in Table XIV, arrows point from specialized goals to more abstract goals. The dotted arrow between goals $O_{520.4}$ and $O_{520.2}$ indicates a conflict inferred from the conflicting deontic modalities "must" and "is not required to."

In goal-oriented requirements engineering, the terms "refinement" and "reduction" are often used to define a relationship between two goals, visualized vertically, in which a high-level goal describes a strategic objective and a low-level goal describes a technical requirement [Antón and W.M. McCracken 1994; van Lamsweerde 2001]. This relationship is equivalent to the roles `isInstrumentOf` and `isPurposeOf` presented in Section 4.2.4, in which the role fillers are high-level and low-level goals,

respectively. This refinement relationship is advantageous to requirements and software engineers because it formalizes the high-level rational for integrating low-level goals into software specifications. On the other hand, the specialization or subsumption hierarchy in Table XIV denotes a different relationship between goals. In a specialization hierarchy, high-level goals *generalize* low-level goals; meaning, all goals that correspond to a low-level goal description can also be described by anyone of the corresponding high-level goal descriptions in their ancestry by following the directed arrows. The specialization hierarchy is advantageous to engineers because any pre- and post-conditions and other relevant context that applies to high-level goals may also apply to low-level goals in the hierarchy. Similarly, goals that appear in conflict such as $O_{520.4}$ and $O_{520.2}$ in Table XIV must be distinguishable by their pre- and post-conditions to ensure these goals are properly achieved in their respective contexts. These specialization relationships may not be obvious to engineers based solely on the wording of a large policy or regulatory document.

Description Logic is necessary but not sufficient to define a complete, comparable semantics for goals. We identified several phrases in the health care and finance domains that required an extended semantics to express arithmetic [Breaux and Antón 2005a; 2005c], deontic [Breaux and Antón 2005a; Breaux et al. 2006] and temporal constraints [Breaux and Antón 2005a] among goals. Arithmetic and comparative constraints restrict the measurable values of properties, such as the age of people by using the phrases "younger than" or "older than". Deontic constraints are mapped from modal phrases such as "may", "must," "shall", and "must not" to predicates in Deontic Logic to infer "what is permissible," called permissions, and

"what ought to be," called obligations [Horty 2001]. Similarly, certain words and verb tenses can be mapped into Temporal Logic to express the relative time-order of events, including which events occur "before," "during" and "after" other events. Further work is needed to understand the complexity and necessity of reasoning using hybrid logics to support the software requirements engineering effort.

Although some might think that Semantic Parameterization requires extensive linguistic knowledge, the dictionary, phrase heuristics and NL patterns require only an elementary understanding of English grammar. However, knowledge of DL is required and poses a limitation to the practical application of this process. Furthermore, there are always questions that must be assumed away for the sake of concentrating on one issue, such as the formalization of domain descriptions in terms of concepts and roles. The parameterization process demonstrates the challenge faced by requirements engineers and provides significant guidance with this aspect of the formalization continuum. Furthermore, the widespread success of this process will depend upon the eventual improvement and applicability of natural language techniques to requirements engineering. This work is an attempt to provide a concrete context and set of problems to which future NL techniques can be applied. Finally, to date we have validated the applicability of Semantic Parameterization to information systems in healthcare and finance. The extent to which this process is applicable to embedded systems or even decision support systems has not been validated.

As with other formal methods and with regards to natural language processing in general, it is important to acknowledge that no matter how rigorous and correct,

any formal manipulation of an inaccurate representation will only produce more inaccurate representations. While these methods help engineers to check their assumptions, domain experts must still evaluate the accuracy of the representation. Consequently, the ability to present domain experts with multiple views of a single representation, obtained through such manipulation, can help detect errors in the representation and improve the resulting requirements specifications. Semantic Parameterization provides one means to obtain such views by enabling engineers and domain experts to query and present formal representations of goals in specialization hierarchies.

In this paper, we presented the Semantic Parameterization process to support engineers in mapping domain descriptions to formal models expressed in Description Logic. The process exposes four types of ambiguity in domain descriptions, including polysemy, synonymy, anaphora or cataphora and under-specifications or omissions. We provided several natural language patterns that are intended to assist engineers with consistently and more efficiently mapping descriptions to formal models. To date, we have developed tools to support the above techniques and to identify ambiguities [Breaux et al. 2006], infer implied stakeholder rights and obligations from parameterized goals [Breaux et al. 2006] and generate domain descriptions from goal models [Breaux and Antón 2005a]. We are currently integrating these techniques into a unified framework. Finally, we summarized the empirical results of applying this process to three studies to demonstrate repeatability and illustrate example applications based on the derived formal specifications.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation ITR Grant: *Encoding Rights, Permissions and Obligations: Privacy Policy Specification and Compliance* (NSF 032-5269); ITR CyberTrust Grant: *Policy-Driven Framework for Online Privacy Protection* (NSF 043-0166); the IBM PhD Fellowship awarded by the Center for Advanced Studies, Research Triangle Park, NC; and CERIAS at Purdue University.

REFERENCES

Antón, A. 1996. Goal-based requirements analysis. In *Proc. IEEE 2nd Int'l Conf. on Requirements Engineering*. IEEE Computer Society, Washington, D.C., 136–144.

Antón, A. and Earp, J. 2004. A requirements taxonomy for reducing web site privacy vulnerabilities. *Requirements Engineering 9,* 3, 169–185.

Antón, A., Earp, J., Bolchini, D., He, Q., Jensen, C., and Stufflebeam, W. 2004. The lack of clarity in financial privacy policies and the need for standardization. *IEEE Security and Privacy 2,* 2, 36–45.

Antón, A. and W.M. McCracken, C. P. 1994. Goal decomposition and scenario analysis in business process reengineering. In *Lecture Notes on Computer Science*. Vol. 811. Springer, Berlin, Germany, 94–104.

Baader, F., Calvanese, D., McGuiness, D., Nardi, D., and Patel-Schneider, P. 2002. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, U.K.

Breaux, T. and Antón, A. 2005a. Analyzing goal semantics for rights, permissions and obligations. In *Proc. IEEE 13th Int'l Conf. on Requirements Engineering*. IEEE Computer Society, Washington, D.C., 177–186.

Breaux, T. and Antón, A. 2005b. Deriving semantic models from privacy policies. In *Proc. IEEE 6th Int'l Workshop on Policies for Distributed Systems and Networks*. IEEE Computer Society, Washington, D.C., 67–76.

Breaux, T. and Antón, A. 2005c. Mining rule semantics to understand legislative compliance. In *Proc. ACM Workshop on Privacy in Electronic Society*. ACM Press, New York, NY, 51–54.

Breaux, T., Vail, M., and Antón, A. 2006. Towards compliance: Extracting rights and obligations to align requirements with regulations. In *Proc. IEEE 14th International Conference on Requirements Engineering*. IEEE Computer Society, Washington, D.C., 46–55.

Chen, P.-S. 1983. English sentence structure and entity-relationship diagrams. *Information Sciences 29,* 2-3, 127–149.

Cregan, A., Schwitter, R., and Meyer, T. 2007. Sydney owl syntax - towards a controlled natural language syntax for owl 1.1. In *Proc. 3rd OWL: Experiences and Directions Workshop*. Vol. 258. CEUR-Workshop Proceedings, Tilburg, Netherlands.

Cysneiros, L. and Leite, J. 2004. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Trans. Knowledge and Data Engineering 30,* 5, 328–350.

Dardenne, A., van Lamsweerde, A., and Fickas, S. 1993. Goal-directed requirements acquisition. *Science of Computer Programming 20*, 3–50.

Denger, C., Berry, D., and Kamsties, E. 2003. Higher quality requirements specifications through natural language patterns. In *Proc. IEEE International Conference on Software: Science, Technology and Engineering*. IEEE Computer Society, Washington, D.C., 80–90.

DHHS. 2003a. Fact sheet: Protecting the privacy of patients' health information. Washington D.C.

DHHS. 2003b. Standards for privacy of individually identifiable health information. 45 CFR Part 160, Part 164 Subpart E. In Federal Register, vol. 68, no. 34.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., and Traverso, P. 2004. Specifying and analyzing early requirements in tropos. *Requirements Engineering 9,* 2, 132–50.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. 2002. Sweetening ontologies with dolce. In *13th Int'l Conf. Knowledge Engineering and Knowledge Management*. Vol. 2473. Springer, Berlin, Germany, 166–181.

Gause, D. and Weinberg, G. 1989. *Exploring Requirements: Quality Before Design*. Dorset House Pub., New York, NY.

Glaser, B. and Strauss, A. 1967. *The Discovery of Grounded Theory*. Aldine Publishing Co., Chicago, IL.

Horty, J. 2001. *Agency and Deontic Logic*. Oxford University Press, New York, NY.

Jackson, M. 1997. The meaning of requirements. *Annals of Software Engineering 3*, 5–21.

Jackson, M. and Zave, P. 1993. Domain descriptions. In *Proc. IEEE Int'l Symp. Requirements Engineering*. IEEE Computer Society, Washington, D.C., 56–64.

Kaindl, H. 1996. How to identify binary relationships for domain models. In *Proc. IEEE 18th Int'l Conf. on Software Engineering*. IEEE Computer Society, Washington, D.C., 28–36.

Kaljurand, K. and Fuchs, N. 2006. Bidirectional mapping between owl dl and attempto controlled english. In *Proc. 4th Workshop on Principles and Practice in Semantic Web Reasoning*. Vol. 4187. Springer, Berlin, Germany, 179–189.

Kaljurand, K. and Fuchs, N. 2007. Verbalizing owl in attempto controlled english. In *Proc. 3rd OWL: Experiences and Directions Workshop*. Vol. 258. CEUR-Workshop Proceedings, Tilburg, Netherlands.

Keenan, E. and Faltz, L. 1985. *Boolean Semantics for Natural Language*. D. Reidel Publishing Co., Boston, Massachusetts, Chapter Transitive Adjective Phrases, 190–192.

Konrad, S. and Cheng, B. 2005. Real-time specification patterns. In *Proc. IEEE 27th Int'l Conf. on Software Engineering*. IEEE Computer Society, Washington, D.C., 372–381.

Matuszek, C., Cabral, J., Witbrock, M., and DeOliveira, J. 2006. An introduction to the syntax and content of cyc. In *Proc. AAAI 2006 Spring Symp. Formalizing and Compiling Background Knowledge and Its Application to Knowledge Representation and Question Answering*. AAAI Press, Menlo Park, California, 44–49.

Mylopoulos, J., Borgida, A., and Yu, E. 1997. Representing software engineering knowledge. *Automated Software Engineering 4,* 3, 291–317.

Niles, I. and Pease, A. 2001. Towards a standard upper ontology. In *Proc. 2nd Int'l Conf. Formal Ontology in Information Systems*. ACM Press, New York, NY, 2–9.

Nirenburg, S. and Raskin, V. 2004. *Ontological Semantics*. MIT Press, Cambridge, MA.

Nuseibeh, B. and Easterbrook, S. 2000. Requirements engineering: a roadmap. In *Proc. IEEE Int'l Conf. on Software Engineering*. IEEE Computer Society, Washington, D.C., 35–46.

Overmyer, S., Lavoie, B., and Rambow, O. 2001. Conceptual modeling through linguistic analysis using lida. In *Proc. IEEE 23rd Int'l Conf. on Software Engineering*. IEEE Computer Society, Washington, D.C., 401–410.

Potts, C. 1997. Requirements models in context. In *Proc. IEEE 3rd Int'l Symp. on Requirements Engineering*. IEEE Computer Society, Washington, D.C., 102–104.

Potts, C., Takahashi, K., and Antón, A. I. 1994. Inquiry-based requirements analysis. *IEEE Software 11,* 2, 21–32.

Ramesh, B. and Jarke, M. 2001. Towards reference models for requirements traceability. *IEEE Trans. on Software Engineering 27,* 1, 58–93.

Smith, R., Avrunin, G., Clarke, L., and Osterweil, L. 2002. Propel: An approach supporting property elucidation. In *Proc. IEEE 24th Int'l Conf. on Software Engineering*. IEEE Computer Society, Washington, D.C., 11–21.

van Lamsweerde, A. 2000. Requirements engineering in the year 00: A research perspective. In *Proc. IEEE 22nd Int'l Conf. Software Engineering*. IEEE Computer Society, Washington, D.C., 5–19.

van Lamsweerde, A. 2001. Goal-oriented requirements engineering: A guided tour. In *Proc. IEEE 5th Int'l Conf. Requirements Engineering*. IEEE Computer Society, Washington, D.C., 249–262.

Wasson, K. 2006. A case study in systematic improvement of language for requirements. In *Proc. IEEE 14th Int'l Conf. on Requirements Engineering*. IEEE Computer Society, Washington, D.C., 6–15.

Wasson, K., Knight, J., Strunk, E., and Travis, S. 2003. Tools supporting the communication of critical domain knowledge in high-consequence systems development. In *Proc. 22nd Int'l Conf. Comp. Safety, Reliability and Security*. Vol. 2788. Springer, Berlin, Germany, 317–330.

Welty, C. 2002. Are upper-level ontologies worth the effort? In *8th Int'l Conf. Principles of Knowledge Representation and Reasoning*. AAAI Press, Menlo Park, California.

Zave, P. and Jackson, M. 1993. Conjunction as composition. *ACM Trans. Software Engineering Methodologies 2,* 4, 379–411.

Received September 2007.