Semantic Parameterization: A Process for Modeling Domain Descriptions

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

1. INTRODUCTION

To validate the correctness, consistency and completeness of software requirements and designs, software engineers require a precise understanding of the functions to be performed by the system and the degree to which those functions satisfy software requirements. In addition, requirements engineers must also consider the role of the software system in the broader context of stakeholder goals. Zave and Jackson note that descriptions in requirements engineering are essentially descriptions of the environment in which systems are intended to operate [Zave and Jackson 1997]. They separately distinguish the environment with and without the system, describing the system as a force that exerts control over actions in the environment. Developing formal descriptions of both the system and its environment will reduce stakeholder misconceptions about the role of the system and expose tacit interactions within the environment that are necessary to validate the correctness of system requirements. In the discussion that follows, we use the term *domain* to refer to the environment and *domain description* to refer to a stakeholder's conceptualization or transcription of a domain problem in natural language [Jackson and Zave 1993].

In requirements engineering, goal-based methods such as the Knowledge Acquisition in AutOmated Specification (KAOS) [Dardenne et al. 1993], Tropos [Fuxman et al. 2004] and the Goal-Based Requirements Acquisition Method (GBRAM) [Antón 1996] have been used to model interactions between systems and their environments in the form of stakeholder goals. Goals describe a state to be achieved or maintained by actors performing actions in the environment. KAOS is unique because it provides a formal semantics to express knowledge at three levels of abstraction: the *meta-level* that is domain independent; the *domain-level* that is domain dependent; and the *instance-level* that concerns the real-world operations for a specific problem [Dardenne et al. 1993]. On the other hand, Tropos provides a formal semantics to the i^* framework which concerns a restricted universe of actors, goals, tasks and resources [Fuxman et al. 2004]. While KAOS encourages developing detailed formal models, Tropos and i^* simplifies the goal-modeling effort by focusing on fewer concerns. Unlike KAOS and Tropos, GBRAM provides engineers with guidelines and heuristics to acquire actors, goals and constraints from domain descriptions [Antón 1996].

In KAOS, Tropos and GBRAM, goals are represented by an informal natural language statement that begins with a verb followed by a phrase. In KAOS, the verb is either achieve, cease, maintain, avoid or optimize [Dardenne et al. 1993], and in Tropos the verb or "mode" is either achieve or maintain [Fuxman et al. 2004].

In GBRAM, the set of verbs is extended for particular domains such as privacy [Antón and Earp 2004]. There are no specific guidelines in either framework for structuring the subsequent phrase. For example, in KAOS 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, 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. Because the phrase is informal and unstructured, these questions can only be identified and answered by an engineer who recognizes these ambiguities. Similarly, the goal "GiveExam" in Tropos implies a transaction between two actors without specifying the recipient of the exam.

Although our analysis has focused on KAOS, Tropos and GBRAM, the fact is that formal methods often concern the semantics required to express phenomena in the domain while assuming that the user of the formal method will correctly align the domain description with the model semantics. In most cases, the domain description is either comprised of thoughts in the heads of domain experts or scripted in natural language documents such as scenarios, interviews and policies. In this paper, we present a process called Semantic Parameterization that helps engineers systematically map domain descriptions into first-order logic expressions for use with other formal methods. The process requires engineers to disambiguate domain descriptions by mapping the lexicon to unique concepts in a dictionary. In addition, engineers model domain descriptions as a conjunction of atomic concepts which forces them to express the tacit relationships between concepts. To increase consistency and efficiency, we summarize several natural language patterns that engineers can use to formalize common phrases in domain descriptions. In addition to presenting empirical evaluation of the process in three studies, we illustrate how the new precision in formal goal specifications can be used to query goal repositories and compare goals in conceptual hierarchies.

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 natural language 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 a few techniques to analyze goals enabled by our approach.

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 [Jackson 1997; Wasson et al. 2003]. In this paper, we focus our attention 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 [Fuchs et al. 2005; Konrad and Cheng 2005; Schwitter 2004; Smith et al. 2002] and standard lexicons [Cysneiros and Leite 2004; Kaindl 1996; Overmyer et al. 2001; Wasson et al. 2003]. At this point in time, we are not concerned with automated approaches to natural language processing that emphasize automation at the cost of accuracy for two fundamental reasons: (1) the full scope of natural language is beyond the scope of software domain descriptions; and (2) inaccuracies in requirements are known to cause an exponential increase in development time and cost [Boehm 1981].

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]. Applying Semantic Parameterization to a problem yields an ontology that is limited to the semantics of that problem. In this approach, the requirements engineer disambiguates terms in domain descriptions only to the extent minimally necessary, not concerning themselves with interpretations of these terms in other domains. While this postpones the extra effort to unify these terms with an upper ontology, the engineer may never need to perform that additional work in a single software project. Rather, we see such debates as distracting from our original goals and, alternatively, 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; Smith et al. 2002] and artificial intelligence [Chen 1983; Fuchs et al. 2005; Schwitter 2004] 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 formal method. Similar to templates and grammars, we provide natural language patterns in Section 5 that help engineers restate goal descriptions into Description Logic expressions. These expressions are used to reason about the classification and composition of goal concepts.

In artificial intelligence, there are three approaches to map a subset of the English language to first-order logic, including Attempo Controlled English (ACE) by Fuchs et al. [Fuchs et al. 2005] and Computer-Processable ENGlish (PENG) by Rolf Schwitter [Schwitter 2004], and Entity-Relationship (ER) models [Chen 1983]. PENG provides formal semantics in first-order logic to compose simple sentences from coordinators (and, or) and subordinators (before, after, if-then) [Schwitter 2004]. ACE presents a case-based analysis of natural language structure that includes special consideration for anaphoric references [Fuchs et al. 2005]. Words with an anaphoric function, such as English articles (e.g., this, that, the), refer the reader to a particular thing or instance in the context of a description. Similar to ACE, our approach requires engineers to distinguish shared instances based on anaphoric references. Unlike ACE and PENG, our approach also requires atomicity in mapping nouns, verbs and adjectives to individual concepts in an ontology. Atomicity, combined with support for inferring the relatedness between instances using conceptual hierarchies, enables a richer query environment than ACE or PENG, which we present in Section 6. Alternatively, 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 a "thing" is an entity or a relation between entities, an ambiguity we call the node-edge problem. In addition to resolving this problem, we describe new rules in addition to those identified by Chen in Section 4. Furthermore, in Section 5 we present empirical validation of our approach in three studies.

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 Acquisition 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]. 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. We approach ambiguity from two perspectives: (1) distinguish between synonymy (same meaning) and polysemy (multiple meanings) at both the conceptual and real-world knowledge senses; and (2) identify under-specifications that result from unspecified individuals that are implied by relations to concepts (e.g., the word

"patient" implies a relationship to a hospital, doctor, etc.)

3. SEMANTIC PARAMETERIZATION

Semantic Parameterization is a process to support engineers who map natural language domain descriptions to models expressed in first-order 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 in which stakeholders can use the conveniences of making ambiguous statements about systems while affording the luxury of detecting and resolving such ambiguities. These luxuries are realized in semi-automated procedures that combine tools, knowledge bases and user feedback. The dictionary that is based on Description Logic and presented in this section establishes the foundation for achieving this goal.

(2) Provide automated support for placing natural language-like inquiries across collections of requirements that answer what, how, why, where and when questions [Potts et al. 1994]. More technically, by enabling the comparison of natural language semantics, the query becomes the atomic function on which to build more complex algorithms for analyzing requirements. We validate 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 ambiguities [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 by providing formal semantics to express two types of stakeholder viewpoints for goals, the purpose for performing actions [Breaux and Antón 2005b] and the actor's right or obligation to perform actions [Breaux et al. 2006].

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

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. 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 these constructors from $\mathcal{ALC}$ with role inversion ($\mathcal{I}$) [Baader et al. 2002]. The

ABox contains assertions about individuals in terms of concepts and roles. DL is an intuitive logic for modeling software engineering problems since the DL notions of description and individual align with the object-oriented notions of class and instance, respectively.

For 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` for the individual y who 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 $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Description Logic defines two special concepts $\top$, 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:

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

Lastly, reasoning in the $\mathcal{ALCI}$ family of DL is known to be PSPACE-complete [Baader et al. 2002].

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 verbs; therefore, $\mathcal{W} = \mathcal{N} \cup \mathcal{J} \cup \mathcal{V}$. Adverbs, which appear in domain descriptions, 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 is one of two possible DL descriptions: 1) a concept C; or, if $w \in \mathcal{N}$, the description can be 2) a role $R.C$ for a some concept C. In the case of roles, 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 `isWordOf`$^-$ $\equiv$ `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.

3.2.1 *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, say $\texttt{isPatientOf}_1\texttt{.Doctor}$ or $\texttt{isPatientOf}_2\texttt{.Hospital}$. The first role describes a patient who has been assigned to a doctor and second role describes 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 $\{ R_i \mid (w, R_i) \in \mathcal{D} \text{ for } 1 \leq i \leq n \}$, 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 specific associations.

3.3 Parameterization Process

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

We illustrate the parameterization process with the domain description $\text{UNLS}_{1.0}$, below. 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 further require adding new DL axioms for subsumption, equivalence and disjointness. The process proceeds in three phases: (1) apply phrase heuristics to disambiguate pronouns and identify extensional synonyms; (2) use the dictionary to assign meanings to the words in the domain description to build the grounding; (3) use the dictionary to identify the tacit relationships between concepts implied by the domain description to build the meta-model. We prepared $\text{UNLS}_{1.0}$ by hyphenating nouns in the statement that describe the same concept. For example, the words “access code”

are hyphenated because they refer to a single concept.

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$_{5,4}$.

In the first phase, the engineer 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 as well (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 synonyms 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 existential 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).

For example, in UNLS$_{1.0}$, 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 (the same individual).

In the second phase, the engineer then builds an extension $\mathcal{A}'$ to the ABox $\mathcal{A}$. For each word $w_{x,y}$ in 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, or $R(p_y, p_v)$ for individual p_y, p_v, if A is a role, noting that the individual p_v may be the same individual from another word $w_{u,v}$ in the prepared statement. For example, in 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 $\texttt{isAccessCodeOf}(p_3, p_1)$ where individual p_3 refers to the *access-code* and p_1 refers to the *customer*. However, if the phrase were simply "access-code$_{3.3}$" with no mention of who possesses the access code, we would have added the assertion $\texttt{AccessCode}(p_3)$. After completing phase one, the extension for UNLS$_{1.0}$ is as follows:

$$\mathcal{A}' = \mathcal{A} \cup \{\texttt{Customer}(p_1) \sqcap \texttt{Share}(p_2) \sqcap \texttt{isAccessCodeOf}(p_3, p_1) \sqcap \texttt{Someone}(p_4) \sqcap \texttt{Provider}(p_4)\}$$

Table I. 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 $word_C$ is a $word_R$ of a $word_D$. For some $(word_C, C), (word_R, R.D), (word_D, D) \in \mathcal{D}$ find individuals x, y such that: $\mathcal{A}'' = \mathcal{A}' \cup \mathcal{A} \cup \{C(x) \wedge R(x, y) \wedge D(y)\}$

Table II. 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**$_{1.1}$.

At this point, all of the words in $\text{UNLS}_{1.0}$ 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 I 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 I, $word_C$, $word_D$, 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 map to extensional references in the UNLS and DL formulas. In addition, the engineer may find that the application of heuristic (ii) or (iii) can be restated as heuristic (iv) by substituting the $word_R$ used in heuristic (ii) or (iii) with $word_C$ in heuristic (iv) and finding a new word $word_R$ that satisfies heuristic (iv). Table II shows the results of applying these primitive RNLS heuristics in Table I to $\text{UNLS}_{1.0}$. Each row in Table II consists of: an index to be used in the following discussion; the heuristic applied from Table I; 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 II. However, using the primitive

RNLS heuristic (iv), the engineer may elicit RNLS (1b) and (1c), exploring the relationship of the access code to the customer as either the property or more generally a 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., their 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 the more general RNLS (1a).

RNLSs (2)-(5) are elicited by recognizing that $\text{UNLS}_{1.0}$ describes an implied activity, in which the customer must not share their access code. The implied activity contributes a new individual p_5 to the meta-model. Each word (customer, share, access-code, and someone) relevant to this single 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 II 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 to the extended ABox as follows:

$$\mathcal{A}'' = \mathcal{A}' \cup \mathcal{A} \cup \{\texttt{Activity}(p_5) \sqcap \texttt{hasSubject}(p_5,p_1) \sqcap \texttt{hasAction}(p_5,p_2) \sqcap \texttt{hasObject}(p_5,p_3) \sqcap \texttt{hasTarget}(p_5,p_4) \sqcap \texttt{isRefrainmentOf}\ (p_5,p_1)\}$$

Recall that one goal in the parameterization process is to relate individuals through as sequence of roles to identify the tacit relationships between individuals. In this example, that sequence of roles is: $\texttt{isAccessCodeOf}(p_3,p_1)$, $\texttt{hasSubject}(p_5,p_1)$, $\texttt{hasAction}(p_5,p_2)$, $\texttt{hasObject}(p_5,p_3)$, $\texttt{hasTarget}(p_5,p_4)$, $\texttt{isRefrainmentOf}(p_5,p_1)$. In this application of the parameterization process, the domain description phrase structure that describes an actor who performs an action on an object is generalized and called the basic activity pattern that appears in Section 4.

Like other forms of conceptual modeling, including object-oriented design, the parameterization process requires engineers to investigate and abstract the tacit or implicit relationships within information. Because this process is intensive, engineers should re-use previously identified RNLS patterns, such as those presented in Section 4, and reserve the process for modeling domain descriptions with new phraseologies.

4. RESTRICTED NATURAL LANGUAGE

The Semantic Parameterization process yields re-usable natural language patterns that are realized as simple sentences called Restricted Natural Language Statements (RNLS). RNLS contain exactly one verb but may contain external references to other RNLS through coordinating words. Based on our experience working within

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

Natural Language Statements	
UNLS$_{2.0}$:	The provider promptly updates erroneous information.
RNLS$_{2.1}$:	The provider promptly updates erroneous information.
Expression	**Pattern**

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

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 following RNLS. In Section 5, we provide empirical evidence to show that these patterns describe the majority of natural language semantics across three different case studies.

The RNLS patterns are organized into the following five categories:

(1) Primitive RNLS that the verbs to-be and to-have (presented in Table I)

(2) Basic and extended activities.

(3) References to other activities, including:
   (a) verb phrases masquerading as nouns;
   (b) transitive verbs followed by verb phrases;
   (c) purposes and instruments; and
   (d) constraints on subjects and objects.

In the following discussion, we provide three examples for each natural language pattern: an domain description (UNLS) and the acquired RNLS; the formal model that maps to the RNLS; and the meta-model for the natural language pattern. In the meta-model, we change domain-specific concepts to the generalized concept `Noun` or `Verb` to indicate which dictionary entries can be used 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 fill 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: 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 III 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

Table IV. Verb phrases masquerading as nouns

Natural Language Statements	Expression

UNLS$_{3.0}$: The provider uses patient information for treatment.
RNLS$_{3.1}$: The provider uses patient-information for (**RNLS**$_{3.2}$).
RNLS$_{3.2}$: Someone treats someone.
	`Activity`
	⊓ `hasSubject.Provider`
	⊓ `hasAction.Use`
	⊓ `hasObject.PatientInformation`
	⊓ `hasPurpose.(Treatment`
	⊓ `hasSubject.Someone`
	⊓ `hasAction.Treat`
	⊓ `hasObject.Someone`
	`)`

intersection (`Activity` ⊓ `Prompt`). Likewise, the adjective "erroneous" describes the noun "information" and appears in (`Information` ⊓ `Erroneous`).

The basic activity pattern is extended for special classes of verbs. For example, transactions are activities whose action word implies the role of another actor who is not the actor performing the action but a participant in the action. Transaction verbs include disclose, share, send, rent, etc. For example, the phrase "to send electronic mail" has the action "send" that requires a *target* to whom the object 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 original semantic relationship. We discuss the following RNLS patterns in detail: verb phrases masquerading as nouns; transitive verbs followed by verb phrases; distinguishing nouns by verb phrases; and purposes and instruments.

4.2.1 *Verb phrases masquerading as nouns.* 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 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 IV, the noun "treatment" is expanded RNLS$_{3.1}$ to RNLS$_{3.2}$ with the ambiguous noun "someone" that is conservatively indexed as an extensional polyseme. In the TBox, we ensure the following axiom is defined to complement this pattern: $\mathcal{T} \models$ `Treatment` ≡ `Activity` ⊓ `hasAction.Treat`. We

Table V. 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`
	`)`

frequently encountered this type of intensional knowledge in the HIPAA case study [Breaux et al. 2006] discussed later in Section 5.

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 and object 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 subject from the domain expert.

In UNLS$_{4.0}$ in Table V, 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 resulting DL expression to the role `hasObject` in the formula from RNLS$_{4.1}$.

4.2.3 *Distinguishing nouns by verb phases.* 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, frequently precede these verb phrases in the 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 changing verb tense to present-simple.

In UNLS$_{5.0}$ presented in Table VI, the obligation to notify does not apply to all customers; rather it is limited to those "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

Table VI. Distinguishing nouns 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`
`  ⊓ hasObject.(Customer`
`  ⊓ isSubjectOf.(Activity`
`  ⊓ hasSource.Provider`
`  ⊓ hasAction.Receive`
`  ⊓ hasObject.HealthService`
`  )`
`  )`
	`Activity`
	`  ⊓ hasSubject.Noun`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.(Noun`
	`  ⊓ isSubjectOf.(Activity`
	`  ⊓ hasAction.Verb`
	`  ⊓ hasObject.Noun`
	`  )`
	`  )`

experts to disambiguate the statement.

To derive the DL formulas, we apply the activity pattern to both RNLS$_{5.2}$ and re-topicalize derived formula for the customer by: inverting the role `hasSubject.Customer` to the intersection `(Customer ⊓ isSubjectOf)` with the remaining description of that activity `(Activity ⊓ hasAction ⊓ hasObject)` assigned to the filler of the role `isSubjectOf`. In the DL formula derived from RNLS$_{5.1}$, the role `hasTarget` is filled by the re-topicalized DL formula from RNLS$_{5.2}$, as shown in Table VI.

4.2.4 *The why and how: purpose and instruments.* 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 V, 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 make no assumptions about the implicit subject and instead use the ambiguous noun "someone" as a placeholder. The formula derived from RNLS$_{6.2}$ is assigned to be the filler of the role `hasPurpose` in the formula derived from RNLS$_{6.1}$.

In Table VI, 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 derived DL expression for RNLS$_{7.2}$ is assigned to the filler of the role `hasInstrument` in the expression for RNLS$_{7.2}$.

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

Table VII. 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`
	`)`

Table VIII. 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`
	`)`

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 should remove the superfluous verb "using" since it is implied by the role `hasInstrument` and simply parameterize the noun using the pattern in Section 4.2.1.

5. VALIDATION AND PROCESS EVOLUTION

Semantic Parameterization has 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. Furthermore, every statement in the domain description was parameterized to avoid overlooking limitations in our process. The three studies are identified as follows:

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

Table IX. 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%

(1) **Goals:** A formative study using the most frequent 100 semi-structured goals from over 1200 goals acquired by applying GBRAM to over 100 Internet privacy policies in the finance and healthcare 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, 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].

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 this case the legal language of government regulations.

In Table IX, we present measures to compare the three studies, including the number of words that appear in domain descriptions, grounding and meta-models; the number of formal models and dictionary entries acquired; the number 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 101^{st} 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 phraseology similar to the RNLS patterns, we observe a higher ratio of grounding words to description words at 82% compared to 19% and 27% in the Facts and Rules studies, respectively, which were conducted using unstructed natural language descriptions. The similar phraseology also accounts for the fewer hours spent during RNLS restatement compared to the Facts and Rules studies. The number of case splits refers to the number of new models generated from logical disjunctions. Case-splits are a subtle factor that impacts the real number of goals acquired from a domain description. For example, in the goal description "providers and third-parties must notify patients of their privacy practices," the obligation of the providers and third-parties are 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.

Table X. 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 (4.2.1)	119	54	164
Transitive verbs (4.2.2)	18	14	75
Purposes (4.2.3)	13	23	45
Instruments (4.2.3)	20	3	26
Nouns Distinguished by verb phrases (4.2.4)	20	5	22

For the number of grounding words g and the number of meta-model words m, the percentage of domain knowledge is $d = g \;/\; (g + m)$. The percentage of meta-model reuse is $1 - d$, which is above 50% in each of these three studies.

In Table X, 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 expressiveness of the RNLS patterns to derive models from unstructured natural language domain descriptions. 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 IX and X. Based on our experience in using goal-mining methodologies in requirements engineering [Antón 1996; Antón and Earp 2004; Antón et al. 2004], we believe that Semantic Parameterization is slightly more complex than goal-mining but provides a richer domain description with increased precision. 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

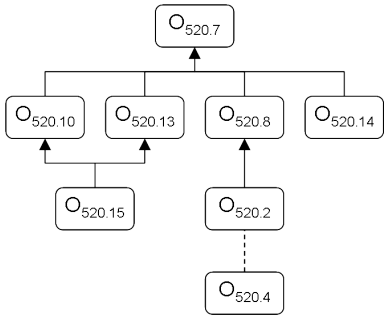
Semantic Parameterization provides new structures to express goals and requirements that have been used to analyze goals, including querying goals by asking open-ended questions [Breaux and Antón 2005a] and organizing goals into abstraction hierarchies [Breaux et al. 2006]. These techniques rely on the fundamental ability to compare goal semantics using the manner in which Semantic Parameterization decomposes goal descriptions into atomic predicates.

Table XI shows the results of a query that was applied to the 100 parameterized goals in the Goals study [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 with whom?" was mapped to the DL expression ($\texttt{Activity} \sqcap \texttt{hasAction.Share} \sqcap \texttt{hasObject.Information} \sqcap \texttt{hasTarget.}\top$) and unified with the knowledge base $\mathcal{KB}$ to obtain the results in Table XI. The first column lists the Goal ID, where duplicate

Table XI. Querying 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

Table XII. Organizing Goals into Conceptual Hierarchies

Goal Description	Goal Hierarchy
$O_{520.2}$: 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.	

IDs indicate a multiple responses due to case splitting from logical disjunctions. The individuals that belong to the roles `isObjectOf.Activity` and `isTargetOf.Activity` answer the question for "what information" and "with whom," respectively.

Table XII shows a set of conceptually similar goal descriptions that were parameterized in the Rules study [Breaux et al. 2006]. The domain description defines a stakeholder 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 in a separate goal hierarchy that compares the goals by each role for subject, action and object of the goal description as well as adverbs and adjectives. In the goal hierarchy in Table XII, arrows point to goals that are conceptually more abstract. The dotted line between goals $O_{520.4}$ and $O_{520.2}$ indicates a conflict inferred from the conflicting deontic modalities "must" and "is not required to."

Description Logic is necessary but not sufficient to define a comparable semantics for goals. Our analysis of goals in the healthcare and finance domains shows a need to express deontic and temporal constraints among goals. The domain description words that indicate deontic constraints such as "may", "must", and "must not"

can be mapped to logical constructors in Deontic Logic to reason about "what is permissible" called rights and "what ought to be" called obligations [Horty 2001]. Similarly, those words and verb tenses that indicate temporal constraints can be mapped to constructors in Temporal Logic to express the 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 combined reasoning using each of these logical formalisms to support the software requirements engineering effort.

To date, we have developed tools to support the above techniques and others 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 and the dictionary into a workbench based on the Eclipse framework. The framework will support the process of creating formal specifications from domain descriptions and the integration of third-party tools to generate partial software specifications from the formal models.

We present the Semantic Parameterization process to support engineers in mapping domain descriptions to formal models. The process exposes ambiguities in domain descriptions by maintaining atomicity among concepts. We provide several natural language patterns that are intended to assist engineers with consistently and more efficiently mapping descriptions to models. Finally, we summarize the empirical results of applying this process to three studies and illustrate example applications based on the 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); 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.

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.

Boehm, B. 1981. *Software Engineering Economics*. Prentice Hall, Upper Saddle River, NJ.

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.

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.

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.

Fuchs, N., Hofler, S., Kaljurand, K., and Rinaldi, F. 2005. Attempo controlled english: A knowledge representation language readable by humans and machines. In *Lecture Notes on Computer Science*. Vol. 3564. Springer, Berlin, Germany, 213–250.

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.

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.

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.

SCHWITTER, R. 2004. Dynamic semantics for a controlled english. In *Proc. IEEE 15th Int'l Workshop on Database and Expert Systems Applications*. IEEE Computer Society, Washington, D.C., 43–47.

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.

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.

ZAVE, P. AND JACKSON, M. 1997. Four dark corners of requirements engineering. *ACM Trans. Software Engineering Methodologies 6,* 1, 1–30.