Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation

Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand
SnT Centre for Security, Reliability and Trust
University of Luxembourg, Luxembourg
{firstname.lastname}@uni.lu

Frank Zimmer
SES Techcom
9 rue Pierre Werner
Betzdorf, Luxembourg
frank.zimmer@ses.com

ABSTRACT

Domain modeling is an important step in the transition from natural-language requirements to precise specifications. For large systems, building a domain model manually is a laborious task. Several approaches exist to assist engineers with this task, whereby candidate domain model elements are automatically extracted using Natural Language Processing (NLP). Despite the existing work on domain model extraction, important facets remain under-explored: (1) there is limited empirical evidence about the usefulness of existing extraction rules (heuristics) when applied in industrial settings; (2) existing extraction rules do not adequately exploit the natural-language dependencies detected by modern NLP technologies; and (3) an important class of rules developed by the information retrieval community for information extraction remains unutilized for building domain models.

Motivated by addressing the above limitations, we develop a domain model extractor by bringing together existing extraction rules in the software engineering literature, extending these rules with complementary rules from the information retrieval literature, and proposing new rules to better exploit results obtained from modern NLP dependency parsers. We apply our model extractor to four industrial requirements documents, reporting on the frequency of different extraction rules being applied. We conduct an expert study over one of these documents, investigating the accuracy and overall effectiveness of our domain model extractor.

Keywords

Model Extraction; Natural-Language Requirements; Natural Language Processing; Case Study Research.

1. INTRODUCTION

Natural language (NL) is used prevalently for expressing systems and software requirements [25]. Building a domain model is an important step for transitioning from informal requirements expressed in NL to precise and analyzable specifications [31]. By capturing in an explicit manner the key concepts of an application domain and the relations between these concepts, a domain model serves both as an effective tool for improving communication between the stakeholders of a proposed application, and further as a basis for detailed requirements and design elaboration [19, 27].

MODELS '16, October 02-07, 2016, Saint-Malo, France
© 2016 ACM. ISBN 978-1-4503-4321-3/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2976767.2976769

Depending on the development methodology being followed, requirements may be written in different formats, e.g., declarative "shall" statements, use case scenarios, user stories, and feature lists [25]. Certain restrictions, e.g., templates [5, 32], may be enforced over NL requirements to mitigate ambiguity and vagueness, and to make the requirements more amenable to analysis. In a similar vein, and based on the application context, the engineers may choose among several alternative notations for domain modeling. These notations include, among others, ontology languages such as OWL, entity-relationship (ER) diagrams, UML class diagrams, and SysML block definition diagrams [3, 16].

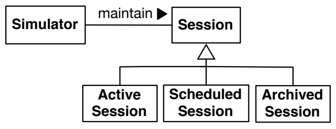
Irrespective of the format in which the requirements are expressed and the notation used for domain modeling, the engineers need to make sure that the requirements and the domain model are properly aligned. To this end, it is beneficial to build the domain model before or in tandem with documenting the requirements. Doing so, however, may not be possible due to time and resource constraints. Particularly, in many industry domains, e.g., aerospace which motivates our work in this paper, preparing the requirements presents a more immediate priority for the engineers. This is because the requirements are a direct prerequisite for the contractual aspects of development, e.g., tendering and commissioning. Consequently, the engineers may postpone domain modeling to later stages when the requirements have sufficiently stabilized and met the early contractual demands of a project. Another obstacle to building a domain model early on in a complex project is the large number of stakeholders that may be contributing to the requirements, and often the involvement of different companies with different practices.

Building a domain model that is aligned with a given set of requirements necessitates that the engineers examine the requirements and ensure that all the concepts and relationships relevant to the requirements are included in the domain model. This is a laborious task for large applications, where the requirements may constitute tens or hundreds of pages of text. Automated assistance for domain model construction based on NL requirements is therefore important.

This paper is concerned with developing an automated solution for extracting domain models from *unrestricted* NL requirements, focusing on the situation where one cannot make strong assumptions about either the requirements' syntax and structure, or the process by which the requirements

R1: The simulator shall maintain the scheduled sessions, the active session and also the list of sessions that have been already handled.

(a)

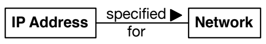


(b)

Figure 1: (a) Example requirements statement and (b) corresponding domain model fragment.

R2: The simulator shall connect only to those **networks for** which the **IP addresses** have been **specified**.

(a)



(b)

Figure 2: (a) Example relative clause modifier (rcmod) dependency and (b) corresponding relation.

were elicited. We use *UML class diagrams* for representing the extracted models. To illustrate, consider requirements statement R1 in Fig. 1(a). This requirement originates from the requirements document of a real simulator application in the satellite domain. Upon manually examining R1, a domain expert sketched the domain model fragment shown in Fig. 1(b). Using heuristic rules implemented via Natural Language Processing (NLP) [18], a tool could automatically identify several of the elements in this model fragment. Indeed, generalizable rules can be provided to extract all the elements, except for Archived Session and its relation to Session, whose identification would require human input.

Automated extraction of models from NL requirements has been studied for a long time, with a large body of literature already existing in the area, e.g., [10, 12, 17, 26, 29, 32], to note some. Nevertheless, important aspects and opportunities that are crucial for the application of model extraction in industry remain under-explored. Notably:

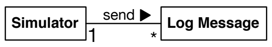
• There is limited empirical evidence about how well existing model extraction approaches perform when applied over industrial requirements. Existing approaches often assume restrictions on the syntax and structure of NL requirements [31]. In many industrial situations, e.g., when there are time pressures or little control over the requirements authoring process, these restrictions may not be met [5]. There is therefore a need to empirically study the usefulness of model extraction over unrestricted NL requirements.

• Modern NLP parsers provide detailed information about the dependencies between different segments of sentences. Our examination of existing model extraction rules indicates that there are important dependency types which are detectable via NLP, but which are not currently being exploited for model extraction. To illustrate, consider requirements statement R2, shown in Fig. 2(a), from the simulator application mentioned earlier. In R2, there is a dependency, called a relative clause modifier (rcmod) dependency [9], between the phrases "network" and "specified". Based on this dependency, which is detected by parsers such as the Stanford Parser [23], one can extract the relation in Fig. 2(b). Existing model extraction approaches do not utilize rcmod and thus do not find this relation. Similar gaps exist for some other dependency types.

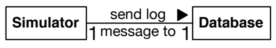
• An important generic class of information extraction rules from the information retrieval literature is yet to be explored for model extraction. This class of rules, referred to as *link paths* [2] (or *syntactic constraints* [13]), enables extracting relations between concepts that are only *indirectly* related. To illustrate, consider requirements statement R3, shown in

R3: The simulator shall send log messages to the database via the monitoring interface.

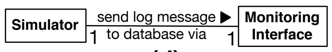
(a)



(b)



(c)



(d)

Figure 3: (a) Example requirements statement, (b) direct relation, (c-d) link path (indirect) relations.

Fig. 3(a), again from the simulator application mentioned earlier. Existing model extraction approaches can detect the relation shown in Fig. 3(b), as "simulator" and "log message" are directly related to each other by being the subject and the object of the verb "send", respectively. Nevertheless, existing approaches miss the indirect relations of Figs. 3(c),(d), which are induced by link paths.

Link path relations can have different *depths*, where the depth represents the number of additional concepts linked to the direct relation. For example, in the relation of Fig. 3(c), one additional concept, namely "database", has been linked to the direct relation, i.e., Fig. 3(b). The depth of the link path relation is therefore one. Using a similar reasoning, the depth of the link path relation in Fig. 3(d) is two.

As suggested by our example, the direct relation of Fig. 3(b) is not the only plausible choice to consider for inclusion in the domain model; the indirect relations of Figs. 3(c),(d) present meaningful alternative (or complementary) relations. Indeed, among the three relations in Figs. 3(b)-(d), the domain expert found the one in Fig. 3(c), i.e., the link path of depth one, useful for the domain model and excluded the other two relations. Using link paths in model extraction is therefore an important avenue to explore.

Contributions. Motivated by addressing the limitations outlined above, we make the following contributions:

(1) We develop an automated domain model extractor for unrestricted NL requirements. We use UML class diagrams for representing the results of extraction. Our model extractor combines existing extraction rules from the software engineering literature with link paths from the information retrieval literature. We further propose new rules aimed at better exploiting the dependency information obtained from NLP dependency parsers.

(2) Using four industrial requirements documents, we examine the number of times each of the extraction rules implemented by our model extractor is triggered, providing insights about whether and to what extent each rule is capable of deriving structured information from NL requirements in realistic settings. The four documents that we consider collectively contain 786 "shall" requirements statements.

(3) We report on an expert review of the output of our model extractor over 50 randomly-selected requirements statements from one of the four industrial documents mentioned above. The results of this review suggest that ≈90% of the conceptual relations identified by our model extractor are either correct or partially correct, i.e., having only minor inaccuracies. Such level of correctness, noting that no particular assumptions were made about the syntax and structure of the requirements statements, is promising. At the same time, we observe that, from the set of relations identified, only ≈36% are relevant, i.e., deemed useful for inclusion in the domain model. Our results imply that low relevance is not a shortcoming in our model extractor per se, but rather a broader challenge to which other rule-based model extractors are also prone. In this sense, our expert

review reveals an important area for future improvement in automated model extraction.

Structure. Sections 2 and 3 review the state of the art and provide background. Section 4 presents our domain model extraction approach. Section 5 reports on an empirical evaluation of the approach. Section 6 concludes the paper.

2. STATE OF THE ART

We synthesize the literature on domain model extraction and compile a set of extraction rules (heuristics) in order to establish the state of the art. The rules identified through our synthesis are shown in Table 1. These rules are organized into four categories, based on the nature of the information they extract (concepts, associations and generalizations, cardinalities, and attributes). We illustrate each rule in the table with an example. We note that the literature provides rules for extracting (class) operations as well. However, and in line with best practice [19], we deem operations to be outside the scope of domain models. Furthermore, since operations typically become known only during the design stages, there is usually little information to be found about operations in requirements documents.

Our focus being on unrestricted NL requirements, we have excluded from Table 1 rules that rely on specific sentence patterns. We do nevertheless include in the table five pattern-based rules (B3 to B5 and D1 to D2) due to the generic nature of these rules. The criterion we applied for the inclusion of a pattern-based rule was that the rule must have been considered in at least two distinct previous publications. We further exclude from Table 1 rules rooted in programming conventions, e.g., the convention of separating concepts and attributes by an underscore, e.g., *Bank_Id*.

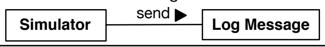
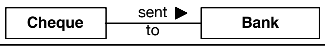
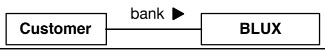
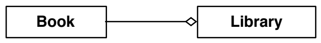
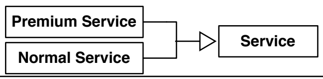
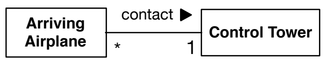
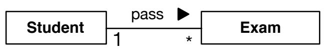
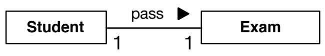
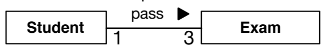
Next, we describe the sources from which the rules of Table 1 originate: The pioneering studies by Abbott [1] and Chen [8] laid the foundation for the subsequent work on model extraction from textual descriptions. Yue et al. [31] survey 20 approaches aimed at transforming textual requirements into early analysis models. Of these, five [4, 15, 20, 21, 24] provide automated support for extracting domain models, or models closely related to domain models, e.g., object models. Yue et al. [31] bring together the rules from the above approaches, further accounting for the extensions proposed to Abbott's original set of rules [1] in other related studies. Rules A1 to A4, B1, B4, B5, C1 to C4, and D1 to D3 in Table 1 come from Yue et al. [31].

To identify more recent strands of related research, we examined all the citations to Yue et al. [31] based on Google Scholar. Our objective was to identify any new extraction rules in the recent literature not already covered by Yue et al. [31]. We found two publications containing new rules: Vidya Sagar and Abirami [29], and Ben Abdessalem Karaa et. al. [7]. Our study of Vidya Sagar and Abirami [29] and a closely-related publication by Elbendak et. al. [12] upon which Vidya Sagar and Abirami build yielded four new rules. These are A5, B2, B3, and D4 in Table 1. As for Ben Abdessalem Karaa et. al. [7], all the new rules proposed therein are pattern-based. These rules do not match our inclusion criterion mentioned above, as no other publication we know of has used these (pattern-based) rules.

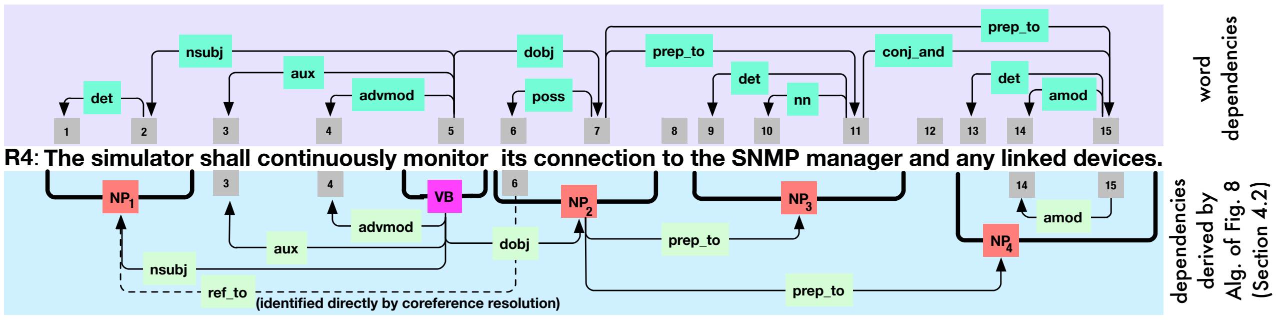
A limitation in the rules of Table 1 is that these rules do not cover link paths, as we already explained in Section 1. Link-path rules have been used in the information retrieval domain for mining structured information from various natural-language sources, e.g., Wikipedia pages [2, 13] and the biomedical literature [30]. However, this class of rules has not been used for model extraction to date. Another limitation, again explained in Section 1, is that existing model extraction rules do not fully exploit the results from NLP tools. Our approach, described in Section 4, proposes extensions in order to address these limitations. Our empirical evaluation, described in Section 5, demonstrates that our extensions are of practical significance.

Further, the large majority of existing work on model extraction is evaluated over exemplars and in artificial settings. Empirical studies on model extraction in real settings remain scarce. Our empirical evaluation, which is conducted in an industrial context, takes a step towards addressing this gap.

Table 1: Existing domain model extraction rules.

	Rule	Description	Example
Concepts	A1	All NPs* in the requirements are candidate concepts.	R3 in Fig. 3 :: **Simulator, Log Message, Database**, and **Monitoring Interface**
	A2	Recurring NPs are concepts.	R3 in Fig. 3 :: **Simulator** (if it is recurring)
	A3	Subjects in the requirements are concepts.	R3 in Fig. 3 :: **Simulator**
	A4	Objects in the requirements are concepts.	R3 in Fig. 3 :: **Log Message**
	A5	Gerunds in the requirements are concepts.	"Borrowing is processed by the staff." :: **Borrowing**
Associations and Generalizations	B1	Transitive verbs are associations.	R3 in Fig. 3 :: 
	B2	A verb with a preposition is an association.	"The cheque is sent to the bank." :: 
	B3	<*R*> in a requirement of the form "<*R*> of <*A*> is <*B*>" is likely to be an association.	"The bank of the customer is BLUX." :: 
	B4	"contain", "is made up of", "include", [...] suggest aggregations / compositions.	"The library contains books." :: 
	B5	"is a", "type of", "kind of", "may be", [...] suggest generalizations.	"Service may be premium service or normal service." :: 
Cardinalities	C1	If the source concept of an association is plural / has a universal quantifier and the target concept has a unique existential quantifier, then the association is many-to-one.	"All arriving airplanes shall contact the control tower." :: 
	C2	If the source concept of an association is singular and the target concept is plural / quantified by a definite article, then the association is one-to-many.	"The student passed the exams." :: 
	C3	If the source concept of an association is singular and the target concept is singular as well, then the association is one-to-one.	"The student passed the exam." :: 
	C4	An explicit number before a concept suggests a cardinality.	"The student passed 3 exams." :: 
Attributes	D1	"identified by", "recognized by", "has" [...] suggest attributes.	"An employee is identified by the employee id." :: ***Employee Id*** is an attribute of **Employee**.
	D2	Genitive cases, e.g., NP's NP, suggest attributes.	"Book's title" :: ***Title*** is an attribute of **Book**.
	D3	The adjective of an adjectivally modified NP suggests an attribute.	"large library" :: ***Size*** is an attribute of **Library**.
	D4	An intransitive verb with an adverb suggests an attribute.	"The train arrives in the morning at 10 AM." :: ***Arrival time*** is an attribute of **Train**.

** NP stands for noun phrase; a definition is provided in Section 3.*



word dependencies
det
nsubj
aux
advmod
dobj
poss
prep_to
det
nn
conj_and
prep_to
det
amod
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
R4: The simulator shall continuously monitor its connection to the SNMP manager and any linked devices.
NP1
3
4
VB
6
NP2
NP3
14
15
NP4
advmod
aux
nsubj
dobj
prep_to
amod
ref_to
(identified directly by coreference resolution)
prep_to
dependencies derived by Alg. of Fig. 8 (Section 4.2)

Figure 5: Results of dependency parsing for requirement R4 of Fig. 4(a).

(a) **R4**: The simulator shall continuously monitor its connection to the SNMP manager and any linked devices.

(b)
```
(ROOT
  (S
    (NP (DT The) (NN simulator))
    (VP (MD shall)
      (ADVP (RB continuously))
      (VP (VB monitor)
        (NP (PRP$ its) (NN connection))
        (PP (TO to)
          (NP
            (NP (DT the) (NNP SNMP) (NN manager))
            (CC and)
            (NP (DT any) (VBN linked) (NNS devices))))))
    (. .)))
```

Figure 4: (a) A requirement and (b) its parse tree.

3. SYNTACTIC PARSING

In this section, we provide background on *syntactic parsing*, also known as syntactic analysis, which is the key enabling NLP technology for our model extraction approach. Syntactic parsing encompasses two tasks: phrase structure parsing and dependency parsing. Our model extractor uses both. We briefly introduce these tasks below.

Phrase structure parsing [18] is aimed at inferring the structural units of sentences. In our work, the units of interest are *noun phrases* and *verbs*. A noun phrase (NP) is a unit that can be the subject or the object of a verb. A verb (VB) appears in a verb phrase (VP) alongside any direct or indirect objects, but not the subject. Verbs can have auxiliaries and modifiers (typically adverbs) associated with them. To illustrate, consider requirements statement R4 in Fig. 4(a). The structure of R4 is depicted in Fig. 4(b) using what is known as a *parse tree*. To save space, we do not visualize the tree, and instead show it in a nested-list representation commonly used for parse trees.

Dependency parsing [28] is aimed at finding grammatical dependencies between the individual words in a sentence. In contrast to phrase structure parsing, which identifies the structural constituents of a sentence, dependency parsing identifies the functional constituents, e.g., the subject and the object. The output of dependency parsing is represented as a directed acyclic graph, with labeled (typed) dependency relations between words. The top part of the graph of Fig. 5 shows the output of dependency parsing over requirements statement R4. An example typed dependency here is nsubj(monitor{5},simulator{2}), stating that "simulator" is the subject of the verb "monitor".

Syntactic parsing is commonly done using the pipeline of NLP modules shown in Fig. 6. In our work, we use the pipeline implementation provided by the Stanford Parser [23]. The first module in the pipeline is the Tokenizer, which splits the input text, in our context a requirements document, into tokens. A token can be a word, a number, or a symbol. The second module, the Sentence Splitter, breaks the text into sentences. The third module, the POS Tagger, attaches a part-of-speech (POS) tag to each token. POS tags represent the syntactic categories of tokens, e.g., nouns, adjectives and verbs. The fourth module, the Named-Entity Recognizer, identifies entities belonging to predefined categories, e.g., proper nouns, dates and locations. The fifth and main module is the Parser, encompassing both phrase structure parsing and dependency parsing. The final module is the Coreference Resolver. This (optional) module finds expressions that refer to the same entity. We concern ourselves with *pronominal* coreference resolution only, which is the task of identifying, for a given pronoun such as "its" and "their", the NP that the pronoun refers to. Fig. 5 shows an example of pronominal coreference resolution, where the pronoun "its" is linked to the referenced NP, namely "the simulator", via a ref_to dependency.

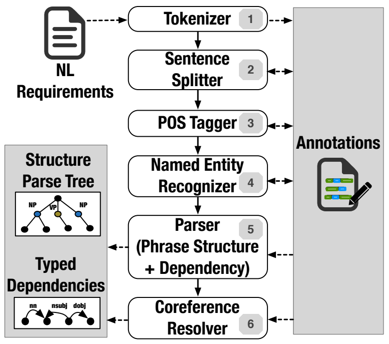


Figure 6: Parsing pipeline.

4. APPROACH

Fig. 7 presents an overview of our domain model extraction approach. The input to the approach is an NL requirements document and the output is a UML class diagram. Below, we elaborate the three main steps of our approach, marked 1-3 in Fig. 7.

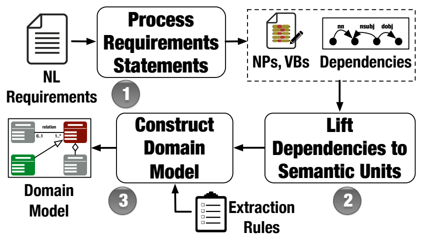


Figure 7: Approach Overview.

4.1 Processing the Requirements Statements

The requirements processing step includes the following activities: (1) detecting the phrasal structure of the requirements, (2) identifying the dependencies between the words in the requirements, (3) resolving pronominal coreferences, and (4) performing stopword removal and lemmatization; these are common NLP tasks, respectively for pruning words that are unlikely to contribute to text analysis, and for transforming words into their base morphological form.

Activities (1), (2), and (3) are carried out by the pipeline of Fig. 6. From the parse tree generated by this pipeline for each requirements statement, we extract the *atomic* NPs and the VBs. Atomic NPs are those that cannot be further decomposed. For example, from the parse tree of Fig. 4(b), we extract: “The simulator” (NP), “monitor” (VB), “its connection” (NP), “the SNMP manager” (NP) and “any linked devices” (NP). We do not extract “the SNMP manager and any linked devices” because this NP is not atomic.

We then subject the NPs to stopword removal. Stopwords are words that appear so frequently in the text that they no longer serve an analytical purpose [22]. Stopwords include, among other things, determiners and predeterminers. In our example, stopword removal strips the extracted NPs of the determiners “the” and “any”. The VBs and the tail words of the NPs are further subject to lemmatization. In our example, “linked devices” is transformed into “linked device”. Had the VB been, say, “monitoring”, it would have been transformed into “monitor”. VBs in passive form are an exception and not lemmatized, e.g., see the example of Fig. 2.

The NPs and VBs obtained by following the process above provide the initial basis for labeling the concepts, attributes, and associations of the domain model that will be constructed in Step 3 of our approach (see Fig. 7). The dependencies obtained from executing the pipeline of Fig. 6 need to undergo further processing and be combined with the results of coreference resolution before they can be used for domain model construction. This additional processing is addressed by Step 2 of our approach, as we explain next in Section 4.2.

4.2 Deriving Dependencies at a Semantic Level

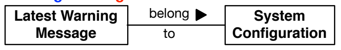
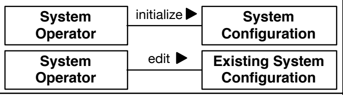
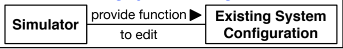
As seen from Fig. 5, the results of dependency parsing (top of the figure) are at the level of words. Many of these dependencies are meaningful for model extraction only at the level of NPs, which, along with verbs, are the main semantic (meaning-bearing) units of sentences. For example, consider the dependency prep_to(connection{7}, manager{11}), stating that “manager” is a prepositional complement to “connection”. To derive from this dependency a meaningful relation for the domain model, we need to raise the dependency to the level of the involved NPs, i.e., prep_to(NP_2, NP_3) in Fig. 5. We do so using the algorithm of Fig. 8.

The algorithm takes as input a set P composed of the atomic NPs and the VBs, and the results of dependency parsing and coreference resolution, all from Step 1 of our approach (Section 4.1). The algorithm initializes the output (i.e., the semantic-unit dependencies) with the ref_to dependencies (L.1), noting that the targets of ref_to dependencies are already at the level of NPs. Next, the algorithm identifies, for each word dependency, the element(s) in P to which the source and the target of the dependency belong (L.3–12). If either the source or target words fall outside the boundaries of the elements in P, the words themselves are treated as being members of P (L.6,11). This behavior serves two purposes: (1) to link the VBs to their adverbial modifiers, illustrated in the example of Fig. 5, and (2) to partially compensate for mistakes made by phrase structure parsers, which are typically only ≈90% accurate in phrase detection [6, 33]. Dependencies between the constituent words of the same NP are ignored (L.13), except for the adjectival modifier (amod) dependency (L.16–18) which is used by rule D3 of Table 1. When the algorithm of Fig. 8 is executed over our example requirements statement R4, it yields the dependencies shown on the bottom of Fig. 5. These dependencies are used in Step 3 of our approach, described next, for extracting associations, aggregations, generalizations and also for linking attributes to concepts.

```
Input: A set $P$ of all (atomic) NPs and VBs in a sentence $S$;
Input: A set $D_{Word}$ of word dependencies in $S$;
Input: A set $R$ of ref_to dependencies for the pronouns in $S$;
Output: A set $D_{Sem}$ of semantic-unit dependencies for $S$;
 1: $D_{Sem} \leftarrow R$;   /*Initialize $D_{Sem}$ with the results of coref resolution.*/
 2: for all $dep \in D_{Word}$ do
 3:    if (there exists some $p \in P$ to which $dep.source$ belongs) then
 4:       $p_{source} \leftarrow p$;
 5:    else
 6:       $p_{source} \leftarrow dep.source$;
 7:    end if
 8:    if (there exists some $p \in P$ to which $dep.target$ belongs) then
 9:       $p_{target} \leftarrow p$;
10:    else
11:       $p_{target} \leftarrow dep.target$;
12:    end if
13:    if ($p_{source} \neq p_{target}$) then
14:       Add to $D_{Sem}$ a new dependency with source $p_{source}$,
          target $p_{target}$ and type $dep.type$;
15:    else
16:       if ($dep.type$ is amod) then
17:          Add $dep$ to $D_{Sem}$;
18:       end if
19:    end if
20: end for
21: return $D_{Sem}$
```

Figure 8: Algorithm for lifting word dependencies to semantic-unit dependencies.

Table 2: New extraction rules in our approach.

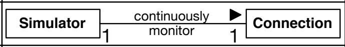
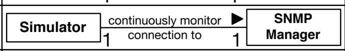
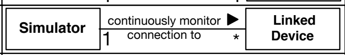
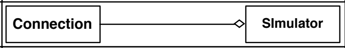
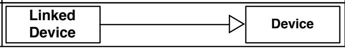
Rule	Description	Example
N1	Relative clause modifiers of nouns (*rcmod* dependency) suggest associations.	“The system operator shall display the **system configuration**, **to** which the **latest warning message belongs**.”  Latest Warning Message belong ▶ to System Configuration *(Another example for rcmod was given in Fig. 2)*
N2	Verbal clausal complements (*ccomp/xcomp* dependencies) suggest associations.	“The **system operator** shall be able to **initialize** the **system configuration**, and to **edit** the **existing system configuration**.”  System Operator initialize ▶ System Configuration System Operator edit ▶ Existing System Configuration
N3	Non-finite verbal modifiers (*vmod* dependencies) suggest associations.	“The **simulator** shall **provide a function to edit** the **existing system configuration**.”  Simulator provide function ▶ to edit Existing System Configuration

4.3 Domain Model Construction

The third and final step of our approach is constructing a domain model. This step uses the NPs and VBs identified in Step 1 (after stopword removal and lemmatization), along with the semantic-unit dependencies derived in Step 2. The extraction rules we apply for model construction are: (1) the rules of Table 1 gleaned from the state of the art, (2) three new rules, described and exemplified in Table 2, which we propose in order to exploit dependency types that have not been used for model extraction before, and (3) link paths [2], which we elaborate further later in this section. In Table 3, we show all the model elements that our approach extracts from our example requirements statement R4. In the rest of this section, we outline the main technical factors in our domain model construction process. We organize our discussion under five headings: domain concepts, associations, generalizations, cardinalities, and attributes.

Domain concepts. All the extracted NPs (from Step 1) are initially considered as candidate concepts. If a candidate concept appears as either the source or the target of some

Table 3: Extraction results for R4 of Fig. 4(a).

#	Type of element(s) extracted	Extracted element(s)	Extraction rule(s) triggered
1	Candidate Concept	**Simulator**, **Connection**, **SNMP Manager**, **Linked Device, Device**	A1*
2	Association		B1, C3 (for cardinalities)
3	Association	 Simulator continuously monitor connection to 1 1 SNMP Manager	Link Path Depth 1, C3 (for cardinalities)
4	Association	 Simulator continuously monitor connection to 1 * Linked Device	Link Path Depth 1, C2 (for cardinalities)
5	Aggregation		D2† (+ coreference resolution)
6	Generalization		D3†

A1 in this table is an enhanced version of A1 in Table 1, as discussed in Section 4.3.

†*As we explain in Section 4.3, in contrast to some existing approaches, we use D2 and D3 (from Table 1) for extracting aggregations and generalizations, respectively, rather than attributes.*

Table 4: Different subject types.

Subject Type	Example	Subject
Simple Subject	"The operator shall initialize the system configuration."	operator
Passive Subject	"The system configuration shall be initialized by the operator."	operator
Genetive Subject	"The operator of the ground station shall initialize the system configuration."	operator

dependency (from Step 2), the candidate concept will be marked as a domain concept. If either the source or the target end of a dependency is a pronoun, that end is treated as being the concept to which the pronoun refers. Table 1 lists a total of five rules, A1–A5, for identifying domain concepts. Our approach implements A1, which subsumes A2–A5. We enhance A1 with the following procedure for NPs that have an adjectival modifier (amod dependency), as long as the adjective appears at the beginning of an NP after stopword removal: we remove the adjective from the NP and add the remaining segment of the NP as a domain concept. For example, consider row 1 of Table 3. The concept of Device here was derived from Linked Device by removing the adjectival modifier. The relation between Device and Linked Device is established via rule D3 discussed later (see *Generalizations*).

Associations. The VBs (from Step 1) that have subjects or objects or both give rise to associations. The manifestation of the subject part may vary in different sentences. Table 4 lists and exemplifies the subject types that we handle in our approach. Our treatment unifies and generalizes rules B1 and B2 of Table 1. We further implement rule B3, but as we observe in our evaluation (Section 5), this rule is not useful for the requirements in our case studies.

For extracting associations, we further propose three new rules, N1–N3, shown in Table 2. Rule N1 utilizes relative clause modifier (rcmod) dependencies. In the example provided for this rule in Table 2, "system configuration" is modified by a relative clause, "to which the latest warning message belongs". From this, N1 extracts an association between System Configuration and Latest Warning Message.

Rule N2 utilizes verbal clausal complement (ccomp and xcomp) dependencies. In the example given in Table 2, "initialize" and "edit" are clausal complements to "able". Here, the subject, "system operator", is linked to "able", and the two objects, "system configuration" and "existing system configuration", are linked to "initialize" and "edit", respectively. What N2 does here is to infer that "system operator" (conceptually) serves as a subject to "initialize" and "edit", extracting the two associations shown in Table 2.

As for Rule N3, the purpose is to utilize non-finite verbal modifier (vmod) dependencies. In the example we show in Table 2, "edit" is a verbal modifier of "function". We use this information for enhancing the direct subject-object relation between "simulator" and "function". Specifically, we link the object of the verbal modifier, "existing system configuration", to the subject, "simulator", extracting an association between Simulator and Existing System Configuration.

The associations resulting from our generalization of B1 and B2, explained earlier, and from our new rules, N1 to N3, are all subject to a secondary process, aimed at identifying link paths [2]. Intuitively, a link path is a combination of two or more direct links. To illustrate, consider rows 3 and 4 in Table 3. The basis for both of the associations shown in these rows is the direct association in row 2 of the table. The direct association comes from the subject-object relation between NP_1 and NP_2 in the dependency graph of Fig. 5. The association in row 3 of Table 3 is induced by combining this direct association with the dependency $\mathsf{prep\_to}(\mathsf{NP}_2,\ \mathsf{NP}_3)$ from the dependency graph. The association of row 4 is the result of combining the direct association with another dependency, $\mathsf{prep\_to}(\mathsf{NP}_2,\ \mathsf{NP}_4)$. In our approach, we consider all possible ways in which a direct association can be combined with paths of prepositional dependencies ($\mathsf{prep\_}*$ dependencies in the dependency graph).

For extracting aggregations, which are special associations denoting containment relationships, we use rules B4 and D2 from Table 1. With regard to D2, we point out that a number of previous approaches, e.g., [12, 31, 29], have used this rule for identifying attributes. Larman [19] notes the difficulty in choosing between aggregations and attributes, recommending that any entity that represents in the real world something other than a number or a string of text should be treated as a domain concept, rather than an attribute. Following this recommendation, we elect to use D2 for extracting aggregations. Ultimately, the user will need to decide which representation, an aggregation or an attribute, is most suitable on a case-by-case basis.

An important remark about D2 is that this rule can be combined with coreference resolution, which to our knowledge, has not been done before for model extraction. An example of this combination is given in row 5 of Table 3. Specifically, the aggregation in this row is induced by the *possessive* pronoun "its", which is linked to "simulator" via coreference resolution (see the ref_to dependency in Fig. 5).

Generalization. From our experience, we surmise that generalizations are typically left tacit in NL requirements and are thus hard to identify automatically. The main rule targeted at generalizations is B5 in Table 1. This rule, as evidenced by our evaluation (Section 5), has limited usefulness when no conscious attempt has been made by the requirements authors to use the patterns in the rule.

We nevertheless observe that certain generalizations manifest through adjectival modifiers. These generalizations can be detected by rule D3 in Table 1. For example, row 6 of Table 3 is extracted by D3. Like rule D2 discussed earlier, D3 has been used previously for attributes. However, without user intervention, identifying attributes using D3 poses a challenge. To illustrate, consider the example we gave in Table 1 for D3. There, the user would need to provide the attribute name, size. For simple cases, e.g., sizes, colors, shapes and quantities, one can come up with a taxonomy of adjective types and use the type names as attribute names [29]. We observed though that generic adjective types are unlikely to be helpful for real and complex requirements. We therefore elect to use D3 for extracting generalizations. Similar to the argument we gave for D2, we leave it to the user to decide when an attribute is more suitable and to provide an attribute name when this is the case.

Cardinalities. We use rules C1 to C4 of Table 1 for determining the cardinalities of associations. These rules are based on the quantifiers appearing alongside the terms that represent domain concepts, and the singular versus plural usage of these terms. For example, the cardinalities shown in rows 2 to 4 of Table 3 were determined using these rules.

Attributes. We use rules D1 and D4 of Table 1 for extracting attributes. As we discussed above, we have chosen to use rules D2 and D3 of Table 1 for extracting aggregations and generalizations, respectively. With regard to rule D4, we note that one cannot exactly pinpoint the name of the attribute using this rule. Nevertheless, unlike rule D3 which is not applicable without user intervention or an adjective taxonomy, D4 can reasonably guess the attribute name. For instance, if we apply our implementation of D4 to the requirement exemplifying this rule in Table 1, we obtain arrive (instead of arrival time) as the attribute name.

5. EMPIRICAL EVALUATION

In this section, we evaluate our approach by addressing the following Research Questions (RQs):

RQ1. How frequently are different extraction rules triggered? One cannot expect large gains from rules that are triggered only rarely. A rule being triggered frequently is thus an important prerequisite for the rule being useful. RQ1 aims to measure the number of times different extraction rules are triggered over industrial requirements.

RQ2. How useful is our approach? The usefulness of our approach ultimately depends on whether practitioners find the approach helpful in real settings. RQ2 aims to assess through a user study the correctness and relevance of the results produced by our approach.

RQ3. Does our approach run in practical time? Requirements documents may be large. One should thus be able to perform model extraction quickly. RQ3 aims to study whether our approach has a practical running time.

5.1 Implementation

For syntactic parsing and coreference resolution, we use Stanford Parser [23]. For lemmatization and stopword removal, we use existing modules in the GATE NLP Workbench [14]. We implemented the model extraction rules using GATE's scripting language, JAPE, and GATE's embedded Java environment. The extracted class diagrams are represented using logical predicates (Prolog-style facts). Our implementation is approximately 3,500 lines of code, excluding comments and third-party libraries. Our implementation is available at: https://bitbucket.org/carora03/redomex.

5.2 Results and Discussion

Our evaluation is based on four industrial requirements documents, all of which are collections of "shall" requirements written in English. Table 5 briefly describes these documents, denoted Case A–D, and summarizes their main characteristics. We use all four documents for RQ1 and RQ3. For RQ2, we use selected requirements from Case A.

Table 5: Description of case study documents.

Case	Description	# of reqs.	Template used?	% of reqs. complying with template
Case A	Simulator application for satellite systems	158	No	N.A.
Case B	Satellite ground station control system	380	Yes (Rupp's)	64%
Case C	Satellite data dissemination network	138	No	N.A.
Case D	Safety evidence information management system for safety certification	110	Yes (Rupp's)	89%

Cases A–C concern software systems in the satellite domain. These three documents were written by different teams in different projects. In Cases B and D, the requirements authors had made an effort to comply with Rupp's template [25], which organizes the structure of requirements sentences into certain pre-defined slots. The number of template-compliant requirements in these two documents is presented in Table 5 as a percentage of the total number of requirements. These percentages were computed in our previous work [5], where Cases B and D were also used as case studies. Our motivation to use template requirements in our evaluation of model extraction is to investigate whether restricting the structure of requirements would impact the applicability of generic extraction rules, which assume no particular structure for the requirements.

RQ1. Table 6 presents the results of executing our model extractor on Cases A–D. Specifically, the table shows the number of times each of the rules employed in our approach has been triggered over our case study documents. The rule IDs correspond to those in Tables 1 and 2; LP denotes link paths. The table further shows the number of extracted elements for each case study, organized by element types.

As indicated by Table 6, rules B1, C1 to C4, D2, D3, and LP are the most frequently triggered. B1 is a generic rule that applies to all transitive verbs. D2 and D3 address genitive cases and the use of adjectives in noun phrases. These constructs are common in English, thus explaining the frequent application of D2 and D3. We note that, as we explained in Section 4.3, we use D2 and D3 for identifying aggregations and generalizations, respectively.

Link paths, as stated earlier, identify indirect associations. Specifically, link paths build upon the direct associations identified by B1, B2, and N1 to N3. To illustrate, consider the example in Fig. 3. Here, B1 retrieves the direct association in Fig. 3(b), and link paths retrieve the indirect ones in Figs. 3(c),(d). In this example, we count B1 as being triggered once and link paths as being triggered twice.

Rules C1 to C4 apply to all associations, except aggregations. More precisely, C1 to C4 are considered only alongside B1 to B3, N1 to N3, and link paths. For instance, C2 is triggered once and C3 twice for the associations of Figs. 3(b-d).

Our results in Table 6 indicate that B3, B5, and D1 were triggered either rarely or not at all in our case study documents. These rules are based on fixed textual patterns. While the patterns underlying these rules seem intuitive, our results suggest that, unless the requirements authors have been trained a priori to use the patterns (not the case for the documents in our evaluation), such patterns are unlikely to contribute significantly to model extraction.

With regard to link paths and our proposed rules, N1 to N3, in Table 2, we make the following observations: Link paths extracted 47% of the (non-aggregation) associations in Case A, 45% in Case B, 50% in Case C, and 46% in Case D. And, rules N1 to N3 extracted 23% of the associations in Case A, 24% in Case B, 29% in Case C, and 37% in Case D. *These percentages indicate that link paths and our new rules*

Table 6: Number of times extraction rules were triggered and number of extracted elements (per document).

	A1[*]	B1	B2	B3	B4	B5	C1-4	D1	D2	D3	D4	N1	N2	N3	LP	# of concepts	# of attributes[†]	# of (regular) associations	# of aggregations	# of generalizations
Case A	370	139	20	0	24	1	526	0	47	76	4	31	48	42	246	370	4	526	71	77
Case B	620	210	19	0	9	1	730	1	81	89	25	58	69	47	327	620	26	730	90	90
Case C	541	68	17	0	5	2	405	0	85	130	21	77	36	6	201	541	21	405	90	132
Case D	85	40	7	0	2	0	274	0	41	35	15	23	11	68	125	85	15	274	43	35

[*] *A1 in this table is an enhanced version of A1 in Table 1, as discussed in Section 4.3. A1 subsumes A2 to A5 (of Table 1), as noted in the same section.*
[†] *The small number of attributes is explained by our decision to use D2 and D3 (resp.) for extracting aggregations and generalizations instead of attributes, as noted in Section 4.3.*

contribute significantly to model extraction. Assessing the quality of the extracted results is the subject of RQ2.

With regard to whether the use of templates has an impact on the applicability of the generic rules considered in this paper, *the results in Table 6 do not suggest an advantage or a disadvantage for templates, as far as the frequency of the application of the extraction rules is concerned.* We therefore anticipate that the generic rules considered in this paper should remain useful for restricted requirements too. Placing restrictions on requirements may nevertheless provide opportunities for developing additional extraction rules [32]. Such rules would naturally be tied to the specific restrictions enforced and are thus outside the scope of this paper.

RQ2. RQ2 aims at assessing practitioners' perceptions about the correctness and relevance of the results produced by our approach. Our basis for answering RQ2 is an interview survey conducted with the lead requirements analyst in Case A. Specifically, we selected at random 50 requirements statements (out of a total of 158) in Case A and solicited the expert's feedback about the model elements that were automatically extracted from the selected requirements. Choosing Case A was dictated by the criteria we had to meet: To conduct the interview, we needed expert(s) who had UML domain modeling experience and who were further fully familiar with the requirements. This restricted our choice to Cases A and B. Our interview would further require a significant time commitment from the expert(s). Making such a commitment was justified by the expert for Case A only, due to the project still being ongoing.

We collected the expert's feedback using the questionnaire shown in Fig. 9. This questionnaire has three questions, Q1 to Q3, all oriented around the notion of "relation". We define relations to include *(regular) associations*, *aggregations*, *generalizations*, and *attributes*. The rationale for treating attributes as relations is the conceptual link that exists between an attribute and the concept to which the attribute belongs. The notion of relation was clearly conveyed to the expert using a series of examples prior to the interview. Our questionnaire does not include questions dedicated to domain concepts, since, as we explain below, the correctness and relevance of the domain concepts at either end of a given relation are considered while that relation is being examined. During the interview, the expert was asked to evaluate, through Q1 and Q2 in the questionnaire, the individual relations extracted from a given requirements statement. The expert was further asked to verbalize his rationale for the responses he gave to these questions. Once all the relations extracted from a requirements statement had been examined, the expert was asked, through Q3, whether there were any other relations implied by that requirements statement which were missing from the extracted results.

The relations extracted from each requirements statement were presented to the expert in the same visual format as depicted by the third column of Table 3. The extraction rules involved were not shown to the expert. To avoid decontex-

Q1 (asked per relation). Is this relation correct?
❑ Yes ❑ Partially ❑ No

Q2 (asked per relation). Should this relation be in the domain model?
❑ Yes ❑ Maybe ❑ No

Q3 (asked per requirements statement). Are there any other relations that this requirements statement implies? If yes, please elaborate.

Figure 9: Interview survey questionnaire.

tualizing the relations, we did not present to the expert the extracted relations in isolation. Instead, a given requirements statement and all the relations extracted from it were visible to the expert on a single sheet as we traversed the relations one by one and asking Q1 and Q2 for each of them.

Q1 addresses correctness. A relation is deemed correct if the expert can infer the relation by reading the underlying requirements statement. We instructed the expert to respond to Q1 by "Yes" for a given relation, if all the following criteria were met: (1) the concept (or attribute) at each end of the relation is correct, (2) the type assigned to the extracted relation (e.g., association or generalization) is correct, and (3) if the relation represents an association, the label and the cardinalities of the association are correct. The expert was instructed to answer by "Partially" when he saw some inaccuracy with respect to the correctness criteria above, but he found the inaccuracy to be minor and not compromising the meaningfulness of the relation; otherwise, the expert was asked to respond by "No".

The correctness of a relation per se does not automatically warrant its inclusion in the domain model. Among other reasons, the relation might be *too obvious* or *too detailed* for the domain model. Q2 addresses relevance, i.e., whether a relation is appropriate for inclusion in the domain model. The expert was asked Q2 for a given relation only upon a "Yes" or "Partially" response to Q1. If the expert's answer to Q1 was "No", the answer to Q2 was an automatic "No". If the expert had answered Q1 by "Partially", we asked him to answer Q2 assuming that the inaccuracy in the relation had been already resolved.

Finally, Q3 addresses missing relations. A relation is missing if it is identifiable by a domain expert upon manually examining a given requirements statement R, but which is absent from the relations that have been automatically extracted from R. A missing relation indicates one or a combination of the following situations: (1) information that is not extracted due to technical limitations in automation, (2) information that is tacit in a requirements statement and thus inferable only by a human expert, (3) information that is implied by the extracted relations, but which the expert decides to represent differently, i.e., using modeling constructs different than the extracted relations. The expert answered Q3 after having reviewed all the relations extracted from a given requirements statement.

Our interview was split into three sessions, with a period of at least one week in between the sessions. The duration of each session was limited to a maximum of two hours to

Table 7: Correctness and relevance results obtained from our expert interview, organized by extraction rules.

(Relation) Extraction Rule																			
B1				B2				B4				D2				D3			
Q1 (Correctness)																			
Y	P	N	C%	Y	P	N	C%	Y	P	N	C%	Y	P	N	C%	Y	P	N	C%
18	11	0	100%	3	1	0	100%	13	0	4	77%	16	4	2	91%	17	0	6	74%
Q2 (Relevance)																			
Y	M	N	R%	Y	M	N	R%	Y	M	N	R%	Y	M	N	R%	Y	M	N	R%
12	0	17	41%	1	0	3	25%	0	0	17	0%	8	0	14	36%	7	4	12	48%

(Relation) Extraction Rule																			
D4				N1				N2				N3				Link Paths			
Q1 (Correctness)																			
Y	P	N	C%	Y	P	N	C%	Y	P	N	C%	Y	P	N	C%	Y	P	N	C%
0	0	2	0%	2	4	1	86%	10	10	0	100%	5	9	1	93%	42	26	6	92%
Q2 (Relevance)																			
Y	M	N	R%	Y	M	N	R%	Y	M	N	R%	Y	M	N	R%	Y	M	N	R%
0	0	2	0%	3	0	4	43%	7	0	13	35%	7	1	7	53%	26	0	48	35%

Legend Q1 (Correctness): Y Yes P Partially N No C% Correctness (%) Q2 (Relevance): Y Yes M Maybe N No R% Relevance (%)

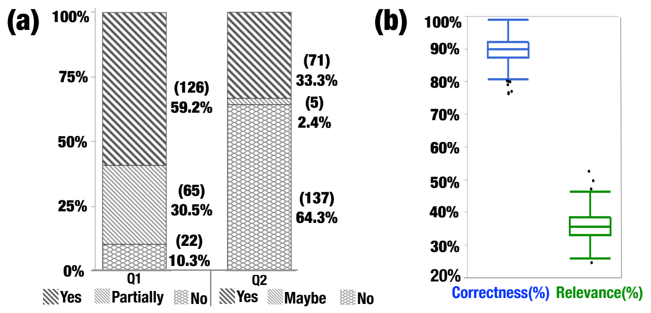


Figure 10: (a) Raw and (b) bootstrapping results for Q1 and Q2.

avoid fatigue effects. At the beginning of each session, we explained and exemplified to the expert the interview procedure, including the questionnaire.

Our approach extracted a total of 213 relations from the 50 randomly-selected requirements of Case A. All these 213 relations were examined by the expert. Fig. 10(a) shows the interview results for Q1 and Q2. As shown by the figure: First, ≈90% of the relations were deemed correct or partially correct, and the remaining 10% incorrect; and second, ≈36% of the relations were deemed relevant or maybe relevant for inclusion in the domain model. The remaining 64% of the relations were deemed not relevant (inclusive of the 10% of the relations that were deemed incorrect).

Due to the expert's limited availability, we covered only ≈32% (50/158) of the requirements statements in Case A. The 213 relations extracted from these requirements constitute ≈31% (213/678) of the total number of relations obtained from Case A by our model extractor. To provide a measure of correctness and relevance which further accounts for the uncertainty that results from our random sampling of the requirements statements, we provide confidence intervals for correctness and relevance using a statistical technique, known as *bootstrapping* [11]. Specifically, we built 1000 resamples with replacement of the 50 requirements that were examined in our interview. We then computed as a percentage the correctness and relevance of the relations extracted from each resample. For a given resample, the correctness percentage is the ratio of correct and partially correct relations over the total number of relations. The relevance percentage is the ratio of relevant and maybe relevant relations over the total number of relations. Fig. 10(b) shows, using box plots, the distributions of the correctness and relevance percentages for the 1000 resamples. *These results yield a 95% confidence interval of 83%–96% for correctness and a 95% confidence interval of 29%–43% for relevance.*

The practical implication of the above findings is that, when reviewing the extracted relations, analysts will have to filter 57%–71% of the relations, despite the large majority of them being correct or partially correct. While we anticipate that filtering the unwanted relations would be more cost-effective than forgoing automation and manually extracting the desired relations from scratch, the required level of filtering needs to be reduced. As we discuss in Section 6, improving relevance and minimizing such filtering is an important direction for future work.

In Table 7, we provide a breakdown of our interview survey results, organized according to the rules that were triggered over our requirements sample and showing the correctness and relevance percentages for each rule. As seen from these percentages, all triggered rules except B4 and D4 proved useful in our study. In particular, the results of Table 7 indicate that our proposed extensions, i.e., rules N1 to N3 and link paths, are useful in practice.

An observation emerging from the relevance percentages in Table 7 (green-shaded cells) is that relevance is low across all the extraction rules and not only for our proposed extensions (N1 to N3 and link paths). *This implies that other existing rule-based approaches for domain model extraction are also susceptible to the relevance challenge.* This observation further underscores the need for addressing the relevance challenge in future work.

As noted earlier, we asked the expert to verbalize his rationale for his responses to Q1 and Q2. This rationale contained a wealth of information as to what made a relation incorrect or only partially correct, and what made a relation not relevant. In Table 8, we provide a classification of the reasons the expert gave for partial correctness and for incorrectness (Q1) and for non-relevance (Q2). The number of relations falling under each category in the classification is provided in the column labeled "Count". For each category, we provide an example of a problematic relation and, where applicable, the relation desired by the expert. The table is self-explanatory. The only remark to be made is that the reasons given by the expert for partial correctness and for incorrectness have one area of overlap, namely *wrong relation type*, as seen from rows 3 and 5 of Table 8. For instance, in the example of row 3, an aggregation was extracted, but the desired relation was an attribute. The expert viewed this inaccuracy as minor. In contrast, in the example of row 5, the expert found the extracted aggregation conceptually wrong, since the desired relation was a generalization.

In response to Q3 (from the questionnaire of Fig. 9), the expert identified 13 missing relations. In six out of these 13 cases, we could automatically extract the missing relation from other requirements statements in our random sample. From the column chart provided for relevance in Fig. 10(a), we see that we have a total of 76 (71+5) relevant and maybe relevant relations that are automatically extracted. This means that our approach automatically retrieved $76/(76 + 7) \approx 92\%$ of the relevant relations.

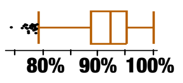
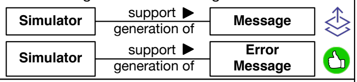
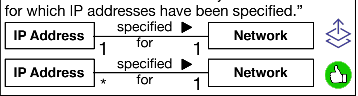
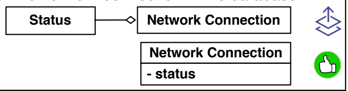
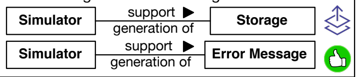
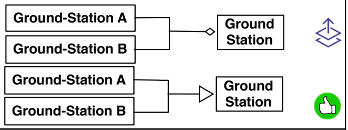
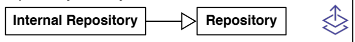
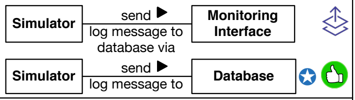
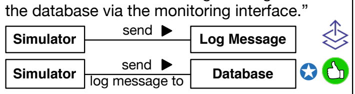
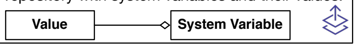


Figure 11: % of relevant relations retrieved.

Using bootstrapping, similar to that done for Q1 and Q2 in Fig. 10(b), we obtain the percentage distribution of Fig. 11 for the retrieved relevant relations. *From*

Table 8: Reasons for inaccuracies and non-relevance.

		#	Reason	Count	Example
Q1	Partially Correct	1	Imprecise label for relation or concept	32	"The simulator shall support the generation of error messages and their storage to the database."  Simulator support ▶ generation of Message Simulator support ▶ generation of Error Message
Q1	Partially Correct	2	Wrong cardinality	29	"The simulator shall connect only to those networks for which IP addresses have been specified."  IP Address specified ▶ for 1 1 Network IP Address specified ▶ for * 1 Network
Q1	Partially Correct	3	Wrong relation type	4	"The system operator shall update the status of the network connection in the database."  Status Network Connection Network Connection - status
Q1	Incorrect	4	Non-existent relation detected	18	"The simulator shall support the generation of error messages and their storage to the database."  Simulator support ▶ generation of Storage Simulator support ▶ generation of Error Message
Q1	Incorrect	5	Wrong relation type	4	"The simulator shall support the simulation of ground stations including Ground-Station A and Ground-Station B."  Ground-Station A Ground-Station B Ground Station Ground-Station A Ground-Station B Ground Station
Q2	Maybe Relevant	6	Future contingency	5	"The simulator shall maintain an internal repository with system variables and their values."  **Expert Feedback:** *All repositories are currently internal. This may change, in which case this relation will be relevant.*
Q2	Not Relevant	7	Relation too detailed	88	"The simulator shall send log messages to the database via the monitoring interface."  Simulator send ▶ log message to database via Monitoring Interface Simulator send ▶ log message to Database
Q2	Not Relevant	8	Incomplete constraint	21	"The simulator shall send log messages to the database via the monitoring interface."  Simulator send ▶ Log Message Simulator send ▶ log message to Database
Q2	Not Relevant	9	Obvious / Common knowledge	6	"The simulator shall maintain an internal repository with system variables and their values." 

✪ *The "desired" relation is indeed also extracted by our model extractor via a different rule. The goal here is to illustrate what the expert considered to be not relevant.*

Legend Extracted Desired

this distribution, we obtain a 95% confidence interval of 82%–100% for the percentage of relevant relations that are automatically extracted by our approach.

RQ3. The execution times for our model extraction approach are in the order of minutes over our case study documents (maximum of ≈4 min for Case B). Given the small execution times observed, we expect our approach to scale to larger requirements documents. Execution times were measured on a laptop with a 2.3 GHz CPU and 8GB of memory.

5.3 Limitations and Validity Considerations

Internal, construct, and external validity are the validity factors most pertinent to our empirical evaluation. With regard to internal validity, we note that our interview considered the correctness and relevance of extracted relations only in the context of individual requirements statements. We did not present to the expert the entire extracted model during the interview. This raises the possibility that the expert might have made different decisions, e.g., regarding the level of abstraction of the domain model, had he been presented with the entire extracted model. We chose to base our evaluation on individual requirements statements, because using the entire extracted model would have introduced confounding factors, primarily due to layout and information overload issues. Addressing these issues, while important, is outside the scope of our current evaluation, whose primary goal was to develop insights about the effectiveness of NLP for domain model extraction. To ascertain the quality of the feedback obtained from the expert, we covered a reasonably large number of requirements (50 requirements, representing nearly a third of Case A) in our interview, and cross-checked the expert's responses for consistency based on the similarities and analogies that existed between the different relations examined.

With regard to construct validity, we note that our evaluation did not include metrics for measuring the amount of tacit expert knowledge which is necessary for building a domain model, but which is absent from the textual content of the requirements. This limitation does not pose a threat to construct validity, but is important to point out in order to clarify the scope of our current evaluation. Building insights about the amount of tacit information that needs to be manually added to the domain model and is inherently impossible to obtain automatically requires further studies.

Finally, with regard to external validity, while our evaluation was performed in a representative industrial setting, additional case studies will be essential in the future.

6. CONCLUSION

We presented an automated approach based on Natural Language Processing for extracting domain models from unrestricted requirements. The main technical contribution of our approach is in extending the existing set of model extraction rules. We provided an evaluation of our approach, contributing insights to the as yet limited knowledge about the effectiveness of model extraction in industrial settings.

A key finding from our evaluation is that a sizable fraction of automatically-extracted relations are not relevant to the domain model, although most of these relations are meaningful. Improving relevance is a challenge that needs to be tackled in future work. In particular, additional studies are necessary to examine whether our observations about relevance are replicable. If so, technical improvements need to be made for increasing relevance. To this end, a key factor to consider is that what is relevant and what is not ultimately depends on the context, e.g., what is the intended level of abstraction, and on the working assumptions, e.g., what is considered to be in the scope of a system and what is not. This information is often tacit and not automatically inferable. Increasing relevance therefore requires a human-in-the-loop strategy, enabling experts to explicate their tacit knowledge. We believe that such a strategy would work best if it is incremental, meaning that the experts can provide their input in a series of steps and in tandem with reviewing the automatically-extracted results. In this way, once a piece of tacit knowledge has been made explicit, it can be used not only for resolving incompleteness in the domain model but also for guiding, e.g., through machine learning, the future actions of the model extractor.

Acknowledgment. We gratefully acknowledge funding from SES and FNR under grants FNR/P10/03 and FNR-6911386.

7. REFERENCES

[1] R. J. Abbott. Program design by informal English descriptions. *Communications of the ACM*, 26(11), 1983.

[2] A. Akbik and J. Broß. Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns. In *Workshop on Semantic Search at the 18th International World Wide Web Conference (WWW'09)*, 2009.

[3] S. Ambler. *The Object Primer: Agile Model-Driven Development with UML 2.0.* Cambridge University Press, 2004.

[4] V. Ambriola and V. Gervasi. On the systematic analysis of natural language requirements with CIRCE. *Automated Software Engineering*, 13(1), 2006.

[5] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE Transactions on Software Engineering*, 41(10), 2015.

[6] G. Attardi and F. Dell'Orletta. Chunking and dependency parsing. In *Workshop on Partial Parsing: Between Chunking and Deep Parsing at 6th International Conference on Language Resources and Evaluation (LREC'08)*, 2008.

[7] W. Ben Abdessalem Karaa, Z. Ben Azzouz, A. Singh, N. Dey, A. S Ashour, and H. Ben Ghazala. Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements. *Software: Practice and Experience*, 2015.

[8] P. P. Chen. English sentence structure and entity-relationship diagrams. *Information Sciences*, 29(2), 1983.

[9] M. C. De Marneffe and C. D. Manning. Stanford typed dependencies manual. Technical report, Stanford University, 2008.

[10] D. K. Deeptimahanti and R. Sanyal. Semi-automatic generation of UML models from natural language requirements. In *4th India Software Engineering Conference (ISEC'11)*, 2011.

[11] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap.* CRC press, 1994.

[12] M. Elbendak, P. Vickers, and N. Rossiter. Parsed use case descriptions as a basis for object-oriented class model generation. *Journal of Systems and Software*, 84(7), 2011.

[13] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Conference on Empirical Methods in Natural Language Processing*, 2011.

[14] GATE NLP Workbench. http://gate.ac.uk/.

[15] H. Harmain and R. Gaizauskas. CM-Builder: A natural language-based CASE tool for object-oriented analysis. *Automated Software Engineering*, 10(2), 2003.

[16] J. Holt, S. Perry, and M. Brownsword. *Model-Based Requirements Engineering.* IET, 2011.

[17] M. Ibrahim and R. Ahmad. Class diagram extraction from textual requirements using natural language processing (NLP) techniques. In *2nd International Conference on Computer Research and Development (ICCRD'10)*, 2010.

[18] N. Indurkhya and F. J. Damerau. *Handbook of natural language processing.* CRC Press, 2010.

[19] C. Larman. *Applying UML and Patterns.* Prentice Hall, 2004.

[20] D. Liu, K. Subramaniam, A. Eberlein, and B. H. Far. Natural language requirements analysis and class model generation using UCDA. In *Innovations in Applied Artificial Intelligence.* Springer, 2004.

[21] D. Liu, K. Subramaniam, B. H. Far, and A. Eberlein. Automating transition from use-cases to class model. In *Canadian Conference on Electrical and Computer Engineering (CCECE'03)*, 2003.

[22] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing.* MIT press, 1999.

[23] M. Marneffe, B. Maccartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *5th International Conference on Language Resources and Evaluation (LREC'06)*, 2006.

[24] L. Mich. NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. *Natural language engineering*, 2(02), 1996.

[25] K. Pohl and C. Rupp. *Requirements Engineering Fundamentals.* Rocky Nook, 2011.

[26] D. Popescu, S. Rugaber, N. Medvidovic, and D. M. Berry. *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs*, chapter Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models. Springer, 2008.

[27] K. Schneider. *Experience and Knowledge Management in Software Engineering*, chapter Structuring Knowledge for Reuse. Springer, 2009.

[28] N. A. Smith. *Linguistic Structure Prediction.* Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2011.

[29] V. B. Vidya Sagar and S. Abirami. Conceptual modeling of natural language functional requirements. *Journal of System and Software*, 88, 2014.

[30] Z. Yang, H. Lin, and Y. Li. BioPPISVMExtractor: A protein–protein interaction extractor for biomedical literature using SVM and rich feature sets. *Journal of biomedical informatics*, 43(1), 2010.

[31] T. Yue, L. Briand, and Y. Labiche. A systematic review of transformation approaches between user requirements and analysis models. *Requirements Engineering*, 16(2), 2011.

[32] T. Yue, L. C. Briand, and Y. Labiche. aToucan: An automated framework to derive UML analysis models from use case models. *ACM Transactions on Software Engineering and Methodology*, 24(3), 2015.

[33] M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu. Fast and accurate shift-reduce constituent parsing. In *51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, 2013.