ABSTRACT
Context. A glossary is an important part of any software requirements document. By making explicit the technical terms in a domain and providing definitions for them, a glossary serves as a helpful tool for mitigating ambiguities.

Goal. A necessary step for building a glossary is to decide upon the glossary terms and to identify their related terms. Doing so manually is a laborious task. Our objective is to provide automated support for identifying candidate glossary terms and their related terms. Our work differs from existing methods for identifying candidate glossary terms and their related terms. Our work differs from existing work on term extraction mainly in that, instead of providing a flat list of candidate terms, our approach clusters the terms by relevance.

Method. We use case study research as the basis for our empirical investigation.

Results. We present an automated approach for identifying and clustering candidate glossary terms. We evaluate the approach through two industrial case studies; one study concerns a satellite software component, and the other— an evidence management tool for safety certification.

Conclusions. Our results indicate that our approach is more accurate than other existing methods for identifying candidate glossary terms; this makes it less likely that our approach will miss important glossary terms. (2) Clustering provides an effective basis for grouping related terms; this makes clustering a useful support tool for selection of glossary terms and associating these terms with their related terms.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: Requirements/Specifications

General Terms
Verification
Keywords
Glossary; Term Extraction; Natural Language Processing (NLP); Clustering; Case Study Research.

1. INTRODUCTION

Natural Language (NL) is commonly used for specifying software requirements. An important consideration for NL requirements is to mitigate ambiguities, i.e., situations where the same requirement may have more than one valid interpretation [20]. Glossaries provide an effective way to improve the understandability of requirements and to reduce the likelihood of different stakeholders developing divergent interpretations of the same requirement. A glossary makes explicit and provides definitions for the technical terms in a domain [20]. A glossary may further provide information about the synonyms, related terms, and example usages of the domain terms. The lack of a glossary can hinder teamwork and potentially jeopardize the success of a project [32].

To ensure that requirements are written using a consistent and unambiguous terminology, it is beneficial to build the glossary at the same time as the requirements are being specified. This, however, is not always feasible due to time pressures. Too much upfront investment into the glossary may also be an issue from a cost-effectiveness standpoint, e.g., when the requirements are volatile and expected to change significantly as they are refined and prioritized.

Consequently, in many industrial settings, requirements glossaries are built after the fact and only when the requirements have sufficiently stabilized. To build a glossary after the fact, analysts necessarily need to go through the underlying requirements documents, label the terms they want to include in the glossary, and further specify any related terms. For large requirements documents, these activities may require a large amount of effort, leaving less human resources for more complex tasks, e.g., writing the definitions for the glossary terms.

Our objective in this paper is to automatically derive candidate glossary terms from NL requirements and to organize these terms into groups of related terms. We illustrate the process using the example in Figure 1: In Figure 1(a), we show the requirements for which a glossary needs to be built. The requirements concern a satellite software component and represent a small fraction of a larger requirements document. The full document is the subject of one of the case studies in this paper, as we discuss later. To protect confidentiality, the requirements in Figure 1(a) have been sanitized without affecting their substance or structure. The abbreviations GSI, STS, and DB in the requirements respectively stand for “Ground Station Interface”, “Surveillance and Tracking System”, and “Database”.

Given the requirements in Figure 1(a), we would like to obtain a set of candidate terms such as those in Figure 1(b). We would then like to group these terms into relevant clusters such as those in Figure 1(c). These clusters, by bringing together relevant information and excluding irrelevant infor-
We propose complementary heuristics to address the limitations we have observed in a naive application of NP chunking to requirements documents.

(2) We devise a technique for clustering candidate glossary terms. Clustering algorithms typically have several parameters that need to be configured for optimal results. One of our key considerations has been to choose a clustering algorithm whose parameters can be set systematically and without resorting to arbitrary rules of thumb. We use the Expectation-Maximization (EM) algorithm [8] for clustering. The EM algorithm minimizes reliance on externally-specified parameters, requiring only the desired number of clusters. For determining this number, precise mathematical guidelines are available. We use the notion of Bayesian Information Criterion (BIC) [24] for determining the optimal number of clusters for a given requirements document.

(3) We report on the design and execution of two industrial case studies. The first study concerns a major proprietary software component in the satellite domain. The requirements document for this component contains 380 requirements statements. The second case study concerns an open-source evidence management system for safety certification. The system is currently under development as part of a large EU project on safety certification, named OPENCOSS (http://www.opencoss-project.eu). The requirements document for the evidence management system has 110 requirements statements.

The results suggest that our approach provides a robust and accurate basis for identification and clustering of candidate glossary terms. To facilitate the replication of our results, we make the material for our second case study available at http://people.svv.lu/arora/esem14.

1.2 Research Questions (RQs)

Our case studies aim at answering the following RQs:

RQ1. How accurate is our approach at identifying glossary terms? A set of candidate terms is accurate if it neither includes too many unwanted terms (false positives) nor misses too many desired terms (false negatives). The aim of RQ1 is to evaluate the accuracy of NP chunking, enhanced with our heuristics, at detecting glossary terms.

RQ2. Which similarity measure for candidate terms yields the most accurate clusters? In order to cluster related terms, one needs a similarity measure to quantify relatedness between pairs of terms. The choice of similarity measure has a major impact on the quality of the resulting clusters. Several alternative measures exist for computing similarities between terms. The aim of RQ2 is to determine which similarity measure leads to the most accurate clusters.

RQ3. How can one systematically specify the number of clusters for the EM algorithm? As stated earlier, to run the EM algorithm, the user needs to specify the number of clusters. Letting the user arbitrarily choose the number can compromise the accuracy of clustering and render the results unpredictable or even useless. The aim of RQ3 is to develop systematic guidelines on how to choose the best number of clusters for a specific requirements document.

RQ4. Is clustering effective at grouping related terms? Clustering is intended at assisting analysts during glossary construction. This is possible only if the generated clusters are sufficiently accurate at grouping related terms. RQ4 provides an initial assessment of the effectiveness of clustering.
RQ5. Does our approach run in reasonable time? One should be able to perform candidate term identification and clustering reasonably quickly, even when faced with a large number of requirements. The aim of RQ5 is to investigate whether our approach has a reasonable running time.

In Section 6, we provide answers to these RQs based on the results of our two case studies.

1.3 Structure of the paper

The rest of the paper is structured as follows: Section 2 compares our work with related work. Section 3 describes our candidate term identification and clustering approach. Section 4 outlines our implementation of the approach. Section 5 presents the design and execution of our case studies. The steps mentioned in this section are exclusively for evaluation of our approach, and are not required in Section 6 reports the results of our empirical validation and answers the RQs that motivate our work. Section 7 discusses threats to validity. Section 8 concludes the paper with a summary and directions for future work.

2. RELATED WORK

Term identification, also called term extraction, has been studied in many domains. Term identification approaches can be broadly classified into three categories [19]: linguistic, statistical, and hybrid approaches. Linguistic approaches, e.g., [6], aim at specifying patterns for detecting terms based on their linguistic properties, e.g., Part-Of-Speech (POS) tags. Statistical approaches, e.g. [13], define algorithms for selection of relevant terms from a list of candidate terms based on statistical measures, e.g., term frequencies. Hybrid approaches, e.g., [5] use a combination of linguistic metadata and statistical measures.

Our approach falls under the linguistic category. The reason why we do not use statistical measures is because these measures are primarily targeted at filtering candidate terms that are likely to be false positives. In terms of classification accuracy metrics, filtering leads to improvements in precision. However, the improvement often comes at a cost to recall, i.e., an increase in false negatives. Further, since all the terms in a requirements document are expected to be used precisely regardless of their statistical characteristics, e.g., the number of times they appear in the document, filtering based on statistical measures is less desirable for requirements documents. In our work, we take a much more conservative approach to filtering and only filter out common nouns (which is a linguistic measure).

Our comparative study for RQ1 (presented in Section 6) compares the accuracy of our approach against several well-known tools, all of which are hybrid. Specifically, we consider the following tools in our comparison:– JATE (Java Automatic Term Extraction toolkit) [33]: JATE implements several term extraction techniques. From the techniques implemented in JATE, we take the one by Frantzi et al. [10]. Among the alternatives in JATE, this technique yields the best accuracy over our case studies. The technique accounts for POS tags, contextual information about the terms, and the frequency and length of the terms.– TextRank [27]: This is a graph-based approach for text processing, with term extraction being one of its constituent parts. Extraction is performed based on POS tags, and an undirected graph in which edges represent pairwise relationships between terms based on their level of cooccurrence.

The approach selects up to a maximum of N terms, where N is specified by the user.

– TOPIA [29]: This is a widely-used python library for term extraction based on POS tags and simple statistical measures, e.g., frequencies.

– TermRaider [26]: This is a recent term extraction tool implemented as a plugin for the GATE NLP Workbench [11]. TermRaider uses advanced heuristics based on POS tags, lemmatization, and statistical measures.

As we argue in Section 6, over requirements documents, our approach yields better recall than all the above, and better precision than all but one (TextRank). Our approach is further distinguished from all the above in that it clusters candidate terms to highlight the relatedness between them.

Term identification has been tackled previously in the field of Requirements Engineering (RE). Aguilera and Berry [1] and Goldin and Berry [12] present frequency-based methods for identifying terms that appear repeatedly in NL requirements. They refer to these terms as “abstractions” which are likely to convey important domain concepts. Zou et al. [35] use automatic term identification in the context of requirements trace retrieval. Their approach uses a POS tagger for extracting single- and double-word noun phrases, and then filters out terms based on frequency measures and certain heuristics, e.g., filtering of single-word terms that also appear within other terms. More recently, Dwarkanath et al. [9] have proposed an approach for identifying candidate glossary terms from requirements based on text parsing and filtering heuristics. These earlier threads, particularly the heuristics proposed in them, have been useful for us to better tune our term identification approach to requirements documents.

The main technical novelty contrasting our approach from the above is the use of clustering for grouping candidate terms. Furthermore, empirical work concerning the application of NLP in the requirements domain remains scarce, particularly in industrial contexts. The case studies we report in this paper are a contribution towards addressing this gap.

This paper uses some of the same NLP technologies that we used in our previous work for checking compliance to requirement templates [4]. We benefit from the results of this earlier work for selecting an effective combination of NLP technologies; however, the NLP technologies per se, in contrast to our earlier work, are neither the main technical component of our approach nor the main focus of our empirical evaluation.

3. APPROACH

Figure 2 shows an overview of our approach. The input to the approach is a set of NL requirements. In the first step, a list of candidate terms is derived from the requirements using Natural Language Processing (NLP). Complementary heuristics are applied in conjunction with NLP for combining and filtering candidate terms. In the second step, a similarity matrix is computed for the candidate terms based on a specific similarity measure. In the third and final step, the candidate terms are clustered based on their similarity degrees. We elaborate these three steps below.

3.1 Identifying Candidate Glossary Terms

The goal of this step is to identify a set of candidate terms. The step is based on an NLP technique called Noun Phrase (NP) chunking [21]. NP chunking processes an input corpus, in this case, one or more requirements documents, and iden-
identifies the sentence segments (chunks) that are NPs. The set of identified NPs constitute the core of the candidate terms.

**Term Identification Heuristics**

We enhance the results of NP chunking via the application of several heuristics. Some of these heuristics aim at re-establishing semantic relationships that are lost during NP chunking. In particular, NP chunking may discard the relationship between a concept and its attributes or sub-parts. For example, chunking the phrase “status of GSI component” would return two NPs: status and GSI component. However, status is unlikely to be useful as a glossary term, unless the context in which it appears is also provided. To capture this intuition, we include as candidate terms phrases of the form: NP of NP, NP’s NP, etc. For example, the term status of GSI component is added to candidate terms through applying the NP of NP heuristic.

We further have a heuristic to probe NPs for the presence of sequences of proper nouns. These sequences are marked as potential keywords along with the enclosing NP. For example, “GSI component” yields GSI as an additional term because GSI is labeled as a proper noun by the POS tagger.

With regards to filtering, as we also stated earlier, we take a conservative approach. We only filter single-word common nouns, e.g. status. By a common noun, we mean a noun that is found in the dictionary, e.g., the WordNet dictionary [30]. Single-word common nouns typically either constitute general knowledge or do not convey any special meaning outside their context. We retain as a candidate term any single-word noun that is not found in the dictionary as well as any single-word common noun that is capitalized – these nouns are likely to denote abbreviations and proper nouns.

Before the application of the second step of our approach, i.e., similarity calculation, all candidate terms are processed and cleared of any determiners, pre-determiners, cardinal numbers, and possessive pronouns. For example, the system operator is reduced to system operator. Furthermore, plural terms are transformed into singular terms using lemmatization, e.g., GSI anomalies is transformed into GSI anomaly. Figure 1(b) shows a list of candidate terms derived from the requirements of Figure 1(a).

### 3.2 Calculating Similarities between Terms

This step is concerned with calculating a similarity matrix to capture the degree of relatedness between every pair of candidate terms. To build this matrix, one may employ two types of similarity measures: *syntactic* and *semantic*.

Syntactic measures are defined over the string content of candidate terms. The measures are typically normalized to produce a value between 0 and 1, with 0 signifying no similarity at all, and 1 signifying perfect similarity, i.e., string equivalence. Numerous techniques exist for measuring similarity between strings. We have experimented with several of these in our study. However, due to limited space and to avoid unnecessary complexity in our experimental results, we cover only the three measures that led to the best clustering results in our case studies (Section 6). These are: (1) *Levenshtein distance* [15], which computes a similarity between two strings based on the minimum number of character edits, i.e., insertions, deletions, and substitutions, required to transform one string into the other. (2) *Monge-Elkan distance* [17], which computes a similarity based on weighted penalties for character insertions and substitutions. (3) *SoftTFIDF* [7], which computes a similarity based on edit distance, combined with the frequency of the single-word constituents of the strings in a corpus. In our case, the corpus is the set of all candidate terms. The intuition here is that two strings are more similar if they share several single-word constituents with comparable frequencies.

Semantic measures, in contrast, capture linguistic correlations between terms. Similar to syntactic measures, semantic measures are often normalized to produce a value between 0 and 1. Here, 0 signifies no linguistic correlation, and 1 signifies equivalence or perfect synonymy. Like syntactic measures, semantic measures have been widely studied. An extensive library of semantic measures is provided in the WordNet::Similarity package [31].

The measures in WordNet::Similarity work only on pairs of single words. To generalize these measures to multi-word terms, we follow the approach of Nejati et al. [18]. Specifically, we treat each candidate term as a set of (single) words. We then use WordNet::Similarity to compute similarities between all pairs of words appearing in the candidate terms. To compute a similarity degree for a given pair of candidate terms, we find an optimal matching between the words of the two terms. A matching is optimal if it maximizes the sum of similarity degrees between the words. We then take the normalized sum associated with an optimal match as the semantic similarity between the two terms.

Semantic measures are best suited for identifying synonyms. These measures cannot be used effectively on their own for calculating similarities between terms [18]. Commonly, semantic measures are combined with syntactic ones. For example, one may take the weighted average or the maximum of syntactic and semantic similarities as the similarity degree for two terms. Using semantic measures is justified only when a reasonable number of synonymous terms are expected in the underlying requirements, e.g., when requirements development is spread over multiple organizations that use different terminologies.

In our case studies (Section 5), we initially combined syntactic and semantic measures by taking the maximum of the two for each pair of terms. We observed that semantic similarities virtually had no impact on the results, due to a near-absence of synonyms in our case studies. Although our implementation (Section 4) supports semantic measures, we do not consider these measures in our evaluation due to limited space and the marginal impact of the measures. Evaluating the usefulness of semantic measures requires further studies where synonymous terms are more frequent.

### 3.3 Clustering

The purpose of this step is to partition the candidate terms into clusters based on the similarity matrix computed in the previous step. Several alternative algorithms exist for clustering. Among them, we choose the Expectation-
Maximization (EM) algorithm [8] as the basis for our work. Our choice is driven by three considerations: (1) EM has only one critical parameter that needs to be specified by the user, namely the number of clusters that the algorithm should produce; (2) One can develop mathematically-based and yet intuitive guidelines on how the number of clusters should be chosen for a specific problem, in our case, a specific requirements document; and (3) the availability of highly-optimized implementations for the EM algorithm in statistical computing tools, e.g., R [28]. This consideration is important to ensure that our approach is scalable.

The EM algorithm is a statistical estimation technique. In the variation of the EM algorithm we use, it is assumed that the observed data, in our case, similarities between candidate terms, is a combination of $N$ multivariate normal distributions. Each distribution, characterized by its mean and covariance matrix, represents one cluster. The goal of the EM algorithm is to approximate the means and covariance matrices of the distributions. It then iterates through the following two steps until convergence:

- **Expectation (E) step:** Given the means and covariance matrices of the $N$ distributions, estimate the membership probability of each candidate term in each distribution.
- **Maximization (M) step:** Estimate new values for the means and covariance matrices of the $N$ distributions, using maximum-likelihood estimation [23].

Once the algorithm converges, each candidate term is assigned to the cluster in which the term has the largest probability of membership. The key remaining question is how to signed to the cluster in which the term has the largest probability.

For identifying candidate terms, we use the OpenNLP Chunker [3]. Our choice to use this chunker is based on a comparative study in our previous work [4], where we show that this chunker is highly accurate and robust over requirements documents. We implement our custom heuristics for combining and filtering noun phrases using the scripting language of the GATE NLP workbench [11]. To compute syntactic similarities, we use a Java-based library, called Simpack [25]. For semantic similarities, we use the WordNet-based implementation by Nejati et al. [18]. EM Clustering and calculation of BIC are done using the mclust package [16] of the R statistical toolkit [28]. For further details see: http://people.svv.lu/arora/esem14.

5. CASE STUDY DESIGN

In this section, we describe the design of our two case studies. The first case study, hereafter called Case-A, concerns a software component being developed at SES TechCom – a leading satellite communication service provider. The second case study, hereafter called Case-B, concerns an evidence management system for safety certification. This system is currently being developed by a consortium of 11 companies and 4 research institutes as part of an EU project named OPENCOSS (http://www.opencoss-project.eu).

5.1 Case Selection

We selected our case studies with the following criteria in mind: (1) We intended to have cases from at least two different domains. In general, conducting multiple case studies is useful for mitigating external validity threats. In our investigation, increasing external validity is particularly important for RQ3, due to the significant impact of the choice of the number of clusters on the effectiveness of our approach. (2) To evaluate the accuracy of our approach, we need a gold standard, i.e., the ideal glossary terms and the ideal clusters of related terms. Building a trustworthy gold standard requires a significant amount of commitment from the experts. Consequently, the availability of experts throughout our investigation was an important consideration. And, (3) we were interested in requirements documents that are reasonably large (> 100 requirements) so as to more conclusively evaluate the scalability of our approach.

The cases we have selected meet the above criteria.

5.2 Data Collection Procedure

Data collection was targeted at building the ideal set of glossary terms and clusters. We elicited the ideal glossary terms directly from the experts. As for the ideal clusters, they were elicited indirectly and through the construction of a domain model. Below, we detail the process for glossary term elicitation and domain model construction. The process for deriving ideal clusters from a domain model is discussed in our analysis procedure (Section 5.3). Note that the domain model and ideal clusters are only for evaluation purposes and not a prerequisite for applying our approach.

**Glossary term elicitation.** For each case, we held walkthrough sessions with the domain experts. In these sessions, the domain experts would first read an individual requirements statement and then identify the glossary terms in that particular statement. Domain experts were asked to identify all the glossary terms in a given statement, irrespective of whether the terms had been already seen in the previous statements. In cases where the domain experts were
Figure 3: Domain model fragment for Case-A

doubtful as to whether a term belonged to the glossary, they were instructed to include the term rather than leave it out, as recommended by glossary construction best practices [20]. The researchers' role in the walkthrough sessions was moderating the sessions and labeling domain experts' choices. Once all the requirements statements were labeled, a duplicate-free list of glossary terms was created. This list is used as the ideal set of glossary terms for our evaluation.

Domain model construction. A domain model is a precise specification of the concepts and associations in a given domain. As we detail in Section 5.3, we use a domain model as an instrument for defining ideal clusters.

The main property we seek in a domain model is for it to be representative of all the terms in a given requirements document. Specifically, we require that any term picked from the requirements document should have a corresponding concept or attribute in the domain model. In Figure 3, we show a (sanitized) fragment of the domain model for Case-A. We use UML class diagrams for capturing domain models, as is common in object-oriented problem analysis [14]. In our domain models, we limit inter-concept associations to specialization (is-a) and aggregation (has-a). These association types characterize the most basic ontological relationships between terms and hence are the most natural to use for defining relatedness between terms.

Each concept or concept attribute x in the domain model is associated with a set, Var(x), of variant terms that are conceptually equivalent to x. For example, consider the status attribute of GSI Component in the model fragment of Figure 3. This attribute is referred to in the requirements document using three variant terms: “status” (where the link to GSI Component is implicit), “status of the GSI Component” and “GSI component status”. To avoid clutter in the figure, we do not show Var(x) when this set has only one term and that term coincides with the name label of x.

Using the notion of Var, the representativeness criterion, mentioned earlier, is formalized as follows: A domain model M is representative if $\bigcup_{x \in M} \text{Var}(x)$ is equal to the set of all candidate terms in the requirements document.

For Case-A, the lead engineer responsible for the requirements (last author) had already built a domain model. The researchers elaborated this model to reach representativeness. A walkthrough of the resulting model was then performed with the lead engineer and the necessary adjustments were made. In Case-B, no domain model existed beforehand. The researchers built one by following standard practices for domain modeling [14]. Subsequently, a walkthrough was held with one of the lead requirements analysts in Case-B to solicit feedback and adjust the model.

5.3 Analysis Procedure

5.3.1 Inferring Ideal Clusters

Ideally, for every candidate glossary term t in a requirements document, we would like to have t and all its “related” terms in one cluster. The main question then is: What does “related” mean? We tackle this question through the observation that behind every requirements document, there is a domain model. This domain model may of course never be built explicitly, or may be partial when it is built. Nevertheless, the observation has useful implications in terms of evaluating our approach. Specifically, given a domain model and a mapping from each concept and attribute of this model onto the terms in the requirements document, one can develop a systematic procedure for inferring ideal clusters. Such a procedure presents two key advantages: First, it alleviates the need for the domain experts to construct the ideal clusters manually – a task that is very laborious for large requirements documents such as the ones in our case studies. Second, although one can never entirely remove subjectivity from how a domain model is constructed and how relatedness is defined, by building an explicit domain model and formulating relatedness in a precise way, one can subject our evaluation process to scientific experimentation.

Equipped with a domain model M, and a function Var(x) for every concept and attribute $x \in M$ (as defined in Section 5.2), we infer the ideal clusters as we outline below.

Ideal clusters are created around concepts, with the concept attributes contributing to some of the clusters. For every concept $c \in M$, we add to the set of ideal clusters one cluster, I, computed as follows: Let $a_1, \ldots, a_n$ denote c's attributes, and let $c_1, \ldots, c_n$ be the set of concepts that are specialized by c (through specialization associations).

$$I = \text{Var}(c) \cup \bigcup_{1 \leq i \leq n} \text{Var}(a_i) \cup \bigcup_{1 \leq j \leq n} \text{Var}(c_j) \cup \bigcup \{\text{Var}(a) | a \text{ is an attribute of some } c; 1 \leq i \leq n\} \cup \bigcup \{\text{Var}(s) | s \text{ is a sibling of } c \text{ via some } c_i; 1 \leq i \leq n\}.$$  

For example, let c be the GSI Anomaly concept in Figure 3. We create a cluster by grouping together the following: c's variant terms (only, "GSI Anomaly"); variant terms for c's attributes (none); variant terms for c's parents ("GSI Monitoring Information") and parents' attributes (none); and variant terms for c's siblings ("GSI Input Parameter" and "GSI Output Parameter").

The above procedure captures relatedness between each concept and its attributes as well as between each concept and other concepts that immediately relate to it via the domain model's inheritance hierarchy. To deal with aggregation associations, we follow a separate procedure: let $c_1$ and $c_2$ be the two ends of an aggregation association. We add one cluster $J = \text{Var}(c_1) \cup \text{Var}(c_2)$ to the set of ideal clusters for each aggregation association.

For example, consider the two aggregation associations in Figure 3. These induce the following clusters: {"Ground Station Interface", "GSI Monitoring Information"} and {"Ground Station Interface", "GSI Component"}.

Our treatment of aggregation associations is motivated by the fact that while an aggregator concept (e.g., Ground Station Interface) is related to each of the concepts it aggregates (aggregatees), there is no relationship between the aggregatees themselves (e.g., GSI Component and GSI Monitoring Information). Hence, putting the aggregatees together into the same cluster, only because they happen to be aggregated by the same concept, does not seem natural.
This argument also extends to concept attributes. While a set of attributes may belong to the same concept, they may be unrelated. The difference though is that attributes, unlike aggregates, have no identity independently of the concept to which they belong. Hence, grouping together all the attributes for a given concept alongside the concept itself, as we do in our approach, seems reasonable.

A final technicality to note is how to handle requirements that concern some, but not all, attributes in a given concept. To illustrate, consider R2 and R6 in Figure 1(a). These requirements envisage that the DB shall store the status and availability attributes of GSI Component, but no such requirement exists for GSI Component’s constraints. In our domain model, we capture the relationship between DB and GSI Component as an aggregation, while keeping track of the specific attributes (here, status and availability) that participate in the relationship. For inferring ideal clusters from such aggregation associations, we use the participating attributes rather than the aggregatee. The motivation for this modeling strategy is to keep the structure of the domain model as simple as possible to avoid understandability issues when validating the model with domain experts.

After creating the ideal clusters in the manner described above, we remove duplicates and any ideal cluster K subsumed by some other ideal cluster K’ (i.e., if K ⊆ K’). We use the resulting clusters as the gold standard for evaluation. The ovals in Figure 7 show the ideal clusters for the example of Figure 1. The way we handle aggregation associations is illustrated by the DB-related clusters in the figure. We discuss Figure 7 further in Section 6 when addressing RQ4.

5.3.2 Evaluation Procedure

There are two main evaluation procedures underlying our empirical results: one for assessing the accuracy of candidate terms and the other for assessing the accuracy of clusters.

Accuracy of candidate terms. We use standard classification accuracy metrics, precision and recall [15], to evaluate the accuracy of candidate terms (step 1 of the approach in Figure 2). The task at hand is to determine which terms belong to the glossary and which terms do not. We use F-measure [15] to combine precision and recall into one metric.

Accuracy of clustering. The choice of metrics for evaluating the accuracy of clustering (step 3 of the approach in Figure 2) is not as straightforward. A wide range of metrics exist to this end, each with its own advantages and limitations. Clusters are typically evaluated across two dimensions: homogeneity and completeness [22]. Homogeneity captures the intuition that the data points (in our case, candidate terms) in a generated cluster should be originating from a single class (i.e., a single ideal cluster). Completeness captures the intuition that all the data points in a class (i.e., an ideal cluster) should be grouped together in one generated cluster. A common limitation of several clustering evaluation metrics, e.g., information-theoretic measures such as entropy, is that they are not particularly suited for situations where the clusters are overlapping [2].

In our work, meaningful handling of overlaps is essential: While our clustering approach (Section 5.3) produces partitions, i.e., non-overlapping clusters, our ideal clusters (Section 5.3) are overlapping. To evaluate the accuracy of clustering, we use a simple set of accuracy metrics – a standard generalization of precision, recall and F-measure for clusters [34] – which can be easily interpreted in the presence of overlaps. Below, we outline the procedure for calculating precision, recall and F-measure for clusters. We discuss the implications of using partition-based clustering in Section 6 when addressing RQ4.

Let I1, . . . , Iℓ denote the set of ideal clusters, and let G1, . . . , Gμ denote the set of generated clusters.

- For every pair (Ii, Gi) 1 ≤ i ≤ t; 1 ≤ j ≤ u:
  - Let ni,j be the number of common data points between Ii and Gi.
  - Precision(Ii, Gi) = \( \frac{n_{i,j}}{|I_i|} \).
  - Recall(Ii, Gi) = \( \frac{n_{i,j}}{|G_j|} \).
  - F-measure(Ii, Gi) = \( \frac{2 \times \text{Precision}(I_i, G_j) \times \text{Recall}(I_i, G_j)}{\text{Precision}(I_i, G_j) + \text{Recall}(I_i, G_j)} \).

- For every ideal cluster Ii, 1 ≤ i ≤ t:
  - Let Gi be the best match for Ii among generated clusters, i.e., F-measure(Ii, Gi) ≥ F-measure(Ii, Gj) for any 1 ≤ j ≤ u. Let Pbest-match(i) denote Precision(Ii, Gi) and let Rbest-match(i) denote Recall(Ii, Gi).
  - Let n = \( \sum_{i=1}^{t} |I_i| \).
  - Compute overall precision and recall as weighted-averages of the precisions and recalls of the ideal clusters:
    - \( \text{Precision} = \sum_{i=1}^{t} \frac{|I_i|}{n} \times P_{\text{best-match}(i)} \).
    - \( \text{Recall} = \sum_{i=1}^{t} \frac{|I_i|}{n} \times R_{\text{best-match}(i)} \).

F-measure for clustering is computed as the harmonic mean of Precision and Recall above. This generalization of classification accuracy metrics for clusters is the basis for answering RQ2 and RQ4 in Section 6.

6. RESULTS AND DISCUSSION

This section describes the case study results for Case-A and Case-B, and discusses the RQs stated in Section 1.2.

The requirements documents in Case-A and Case-B have 380 and 110 requirements statements, respectively. Glossary term elicitation resulted in 140 glossary terms for Case-A and 51 glossary terms for Case-B. Following glossary term elicitation, representative domain models were developed for the two case studies. The domain model for Case-A has 352 classes, 53 attributes, 213 specialization associations, and 95 aggregation associations. The domain model for Case-B has 39 classes, 14 attributes, 5 specialization associations, and 31 aggregation associations. The reason why we have a proportionally small number of attributes in the domain models is because, in line with best practice, when there was uncertainty as to whether an element should be a class or an attribute, we modeled it as a class.

RQ1. The candidate term identification step of our approach (step 1 in Figure 2) yields 644 terms for Case-A and 96 terms for Case-B. Figure 4 shows the classification accuracy results for our approach and compares them against results from four existing term extraction tools, discussed in Section 2. As the chart shows, our approach produces better recall than the four alternatives considered, and better precision than all but TextRank. The deficit in precision when compared to TextRank is small: 2.5% for Case-A and 1% for Case-B, while the gain in recall is large: 22.2% for Case-A and 41.2% for Case-B. We thus conclude that our approach is more accurate than the alternatives considered.

Our approach yields 10 false negatives in Case-A and 5 in Case-B. Of these, 9 are explained by the heuristic we apply for filtering single-word common nouns (Section 3.1). The
remaining 6 are due to mistakes made by the NP chunker. Including single-word common nouns in the candidate terms addresses the first type of false negatives but leaves the analysts with a large number of additional false positives. The latter type of false negatives are unavoidable and arise due to limitations in NLP techniques.

The results in Figure 4 prompted an investigation into why precision for Case-A is low across all approaches, including ours. To identify the cause, we asked the lead engineer in Case-A to explain his rationale in choosing the ideal glossary terms. We determined that his choices exclusively reflected the terms for the glossary of the specific requirements document being analyzed. In other words, he discarded terms which he deemed common knowledge or which he knew were already defined in the glossaries of other related documents.

In general, since contextual factors and working assumptions such as the above are often tacit and thus unavailable to an automated tool, large variations may be seen in terms of precision across different projects.

**RQ2.** The goal of RQ2 is to determine which similarity measure is the most accurate for grouping together glossary terms and their related terms. As suggested by the discussion in RQ1, the set of terms in a requirements document can be wider than what is of interest to the domain experts for glossary construction. To determine which similarity measure produces the best results relevant to the glossary, we need to discard in our analysis the terms that are not relevant. Specifically: (1) we limit ideal clusters to those that contain at least one glossary term or a variation thereof. Let \( C \) denote the set of ideal clusters after applying this restriction and let \( S \) denote the union of the terms in \( C \). (2) From the generated clusters, we prune any term that is not in \( S \).

Note that the generated clusters cover all candidate terms, not only those that are relevant to the glossary. Outside an evaluation setting, one cannot distinguish terms that are relevant to the glossary from those that are not. To preserve the realistic behavior of clustering, it is thus important to compute the generated clusters for all the candidate terms and prune the results, as opposed to narrowing the set of candidate terms to those that are relevant and performing clustering only over relevant terms.

Figure 5 shows the F-measure results for clustering. In our evaluation, we consider only the best three syntactic similarity measures, namely SoftTFIDF, Monge Elkan, and Levenstein. Due to reasons stated in Section 3.2, semantic similarity measures have marginal impact in our case studies and are thus not considered in our evaluation.

In Figure 5, we compute F-measure for all potential numbers of clusters ranging from 1 to \( n/2 \), with \( n \) denoting the total number of candidate terms. The average size of clusters is inversely proportional to the number of clusters. Using a value larger than \( n/2 \) for the number of clusters is unreasonable because the average number of terms in each cluster

**RQ3.** The number of clusters, denoted \( \ell \), is a value between 1 and the total number of candidate terms. For EM clustering to produce useful results, we need to choose a suitable \( \ell \). If \( \ell \) is too small, the results will be poor due to low precision. If \( \ell \) is too large the results will be poor due to low recall. Furthermore, and as discussed in RQ2, considering values larger than half the number of candidate terms for \( \ell \) is unreasonable. Obviously, one does not have access to a gold standard outside an evaluation setting. Therefore, one cannot use the optimal point in an F-measure chart such as the ones in Figure 5 for choosing \( \ell \).

We use BIC (Section 3.3) for choosing \( \ell \). Specifically, we choose an \( \ell \) that maximizes BIC. If the maximum BIC occurs on a plateau, one may choose \( \ell \) based on a slightly smaller BIC from the tail end of the plateau and within a small margin, say 5%, from the maximum BIC. The tail end of the plateau results in a larger number of clusters and hence a smaller number of terms within individual clusters. As long as BIC remains uncompromised, clusters with a smaller number of terms in them are advantageous because such clusters are easier to inspect by analysts.

In Figure 6, we show the BIC charts for Case-A and Case-B. To produce these charts, we have used the SoftTFIDF similarity measure – the measure we observed in RQ2 to be yielding the best results. In Case-A, the maximum BIC occurs when there are 69 clusters. BIC nonetheless stays within a small margin (2.4%) from the maximum up to 120 clusters, at which point a sharp decline is seen. Based on our argument above, we choose \( \ell = 120 \) for Case-A. In Case-B, the maximum BIC occurs at 23 clusters, followed by a steep decline. For this case, we choose \( \ell = 23 \). Although these choices for \( \ell \) do not bring about the absolute best F-measures in the respective case studies, as can be seen from Figure 5, the results are nearly optimal and satisfactory.

**RQ4.** As noted earlier, the ideal clusters in our evaluation are overlapping. The overlaps arise because individual candidate terms assume different roles in the context of different requirements, and thus relate to potentially different terms based on each role. By allowing overlaps in the ideal clusters, one can distinguish different roles and orient each cluster around one particular role. Having the roles separated from one another is advantageous because it makes
### Table 1: Execution times

<table>
<thead>
<tr>
<th>Case / strategy to increment # of clusters</th>
<th>Phase</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-A</td>
<td>Term Identification</td>
<td>10s</td>
</tr>
<tr>
<td></td>
<td>Similarity Calculation</td>
<td>13s</td>
</tr>
<tr>
<td></td>
<td>Clustering</td>
<td>7s</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>29s</td>
</tr>
<tr>
<td>Case-B</td>
<td>Term Identification</td>
<td>72s</td>
</tr>
<tr>
<td></td>
<td>Similarity Calculation</td>
<td>27s</td>
</tr>
<tr>
<td></td>
<td>Clustering</td>
<td>24m</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>35m</td>
</tr>
</tbody>
</table>

The clusters highly-focused and small. This said, overlapping clusters pose a significant overhead, because individual candidate terms can appear multiple times in them. Taking Case-A and Case-B as examples, each candidate term appears approximately twice on average in the ideal clusters. This means that the effort associated with inspecting the ideal clusters will be higher, perhaps twice as high as the case where the clusters are non-overlapping.

Our clustering approach aims to strike a balance between the two inherent tradeoffs above: In our approach, we partition the candidate terms to avoid overhead, but at the same time use overlapping ideal clusters for assessing accuracy.

To illustrate, we show in Figure 7 both the ideal and generated clusters for the example of Figure 1. Here, there are 12 ideal clusters, denoted by ovals, and 7 generated clusters, denoted by colors. The term “status” appears in the ideal clusters but is struck out from the generated ones. This term, which is a variant of the GSI Component’s status attribute in Figure 3, is filtered due to being a single-word common noun (see Section 3.1). The clustering precision and recall for this example are 73.1% and 71.9%, respectively. These numbers mean that, on average, 73.1% of the terms the analyst sees in each generated cluster pertain to the specific aspect of relatedness he is investigating. Further, 71.9% of all the ideal terms for this specific relatedness aspect have been retrieved. Despite the accuracy not being perfect, the generated clusters provide useful cues about the related terms.

Given that achieving perfect accuracy is theoretically impossible (unless the ideal clusters have no overlaps), we need to find an upper bound on the maximum possible accuracy that one can expect from partition-based clustering in our context. This upper bound provides a reference point for evaluating the effectiveness of clustering in our approach.

To compute such an upper bound, we follow a randomized procedure. Specifically, we impose a random order on the ideal clusters and prune these clusters so that the following constraint holds for any given term $t$: if $t$ appears in a cluster $C$ at position $i$ in the ordering, then $t$ cannot appear in any cluster $C'$ at a position $i'$ such that $i' > i$. This procedure derives non-overlapping clusters from the ideal clusters.

The accuracy of these non-overlapping clusters is a good indicator for what partition-based clustering can achieve in the best case. We applied the above procedure to 1000 random orders of the ideal clusters in Case-A and Case-B, and computed the average accuracy measures. For Case-A, the ideal clusters with their overlaps removed yield a precision of 90.8%, recall of 76.2%, and $F$-measure of 82.9%. For Case-B, these numbers are 89.1%, 70.7%, and 78.8%, respectively.

When the best similarity measure identified in RQ2 is used (i.e., SoftTFIDF), and when the number of clusters is chosen based on the BIC value discussed in RQ3, our clustering approach yields the following results: For Case-A, we obtain a precision of 65.5%, recall of 38.9%, and $F$-measure of 48.8%. For Case-B, these numbers are 66.8%, 52.5%, and 58.8%, respectively. While further analysis in the form of user studies is necessary to assess the effectiveness of the generated clusters, we believe these results are reasonable given the positive feedback from the domain experts about the generated clusters, and the accuracy upper bounds computed above.

**RQ5.** Our scalability analysis is concerned with ensuring that the steps of the approach in Figure 2 run within reasonable time. The running time of our approach is dominated by the construction of the BIC curve in the clustering step. To build a BIC curve similar to those in Figure 6, one needs to calculate one BIC value for every possible number of clusters, where instead of calculating BIC for every possible (unless the ideal clusters have no overlaps), we need to find an upper bound on the maximum possible accuracy that one can expect from partition-based clustering in our context. This upper bound provides a reference point for evaluating the effectiveness of clustering in our approach.

To compute such an upper bound, we follow a randomized procedure. Specifically, we impose a random order on the ideal clusters and prune these clusters so that the following constraint holds for any given term $t$: if $t$ appears in a cluster $C$ at position $i$ in the ordering, then $t$ cannot appear in any cluster $C'$ at a position $i'$ such that $i' > i$. This procedure derives non-overlapping clusters from the ideal clusters.

The accuracy of these non-overlapping clusters is a good indicator for what partition-based clustering can achieve in the best case. We applied the above procedure to 1000 random orders of the ideal clusters in Case-A and Case-B, and computed the average accuracy measures. For Case-A, the ideal clusters with their overlaps removed yield a precision of 90.8%, recall of 76.2%, and $F$-measure of 82.9%. For Case-B, these numbers are 89.1%, 70.7%, and 78.8%, respectively.

When the best similarity measure identified in RQ2 is used (i.e., SoftTFIDF), and when the number of clusters is chosen based on the BIC value discussed in RQ3, our clustering approach yields the following results: For Case-A, we obtain a precision of 65.5%, recall of 38.9%, and $F$-measure of 48.8%. For Case-B, these numbers are 66.8%, 52.5%, and 58.8%, respectively. While further analysis in the form of user studies is necessary to assess the effectiveness of the generated clusters, we believe these results are reasonable given the positive feedback from the domain experts about the generated clusters, and the accuracy upper bounds computed above.
terms, hence, a 1% increment is 6.44 terms. We thus calculate BIC at 1, [6.44], [6.44 * 2], and so on. This strategy yields a constant number of points (100 points) on the X axis, irrespective of the actual number of terms. Using this strategy, we reduce the clustering running time for Case-A to 24 minutes. If necessary, one can further reduce the running time by considering less points for BIC calculations or less probabilistic model alternatives for the EM algorithm.

7. THREATS TO VALIDITY

Internal validity. Learning is a potential threat to the internal validity of our work. We mitigated this threat by (1) avoiding the exposure of the domain experts, during data collection, to the generated clusters, and (2) refraining from using case study data as test data for tool development.

Construct validity. The definition of ideal clusters is a subjective one, as explained in Section 5.3. To mitigate construct validity threats, we applied an explicit and systematic process for defining the ideal clusters, building on the notion of a domain model. This limits subjectivity in defining the ideal clusters and further makes the process repeatable.

Conclusion validity. The main factor to discuss in relation to conclusion validity is the choice of accuracy metrics used for evaluating generated clusters. We chose the metrics in our evaluation to best match the overlapping nature of the ideal clusters in our problem. This helps mitigate threats to conclusion validity. Future user studies are necessary to ascertain whether the metrics we use are adequate.

External validity. We applied our approach to two industrial case studies from two different domains. The consistency seen across the results of the two studies helps build confidence about the generalizability of our approach. Further case studies, particularly over requirements documents written by multiple organizations, are nonetheless necessary for improving external validity.

8. CONCLUSION

We presented an automated approach for identifying and clustering candidate glossary terms. We reported on the application of our approach to two industrial case studies, in the context of which we evaluated the accuracy of the approach. One of the main advantages of our approach is that it comes equipped with guidelines on how to tune clustering for a given requirements document. This is important for a successful application of the approach outside our case studies. In future work, we plan to investigate ways to further increase the accuracy of our approach through semantic analysis of requirements statements and more tailored strategies for clustering. We further plan to include an interactive feedback loop in our approach so that we can use the decisions already made by the experts for providing better recommendations for subsequent decisions. Lastly, we need to conduct more empirical studies to provide a definitive picture about the approach’s practical utility, particularly in comparison with manual elicitation of glossary terms.

Acknowledgments. Funding was provided by FNR under grants FNR/P10/03 and FNR6911386. We thank Sunil Nair and Jose L. de la Vara for helping with our second case study.

9. REFERENCES


