CMT and FDE: Tools to Bridge the Gap between Natural Language Documents and Feature Diagrams

Alessio Ferrari
ISTI-CNR, Pisa, Italy
alessio.ferrari@isti.cnr.it

Giorgio O. Spagnolo
ISTI-CNR, Pisa, Italy
spagnolo@isti.cnr.it

Stefania Gnesi
ISTI-CNR, Pisa, Italy
stefania.gnesi@isti.cnr.it

Felice Dell'Orletta
ILC-CNR, Pisa, Italy
felice.dellorletta@ilc.cnr.it

ABSTRACT

A business subject who wishes to enter an established technological market is required to accurately analyse the features of the products of the different competitors. Such features are normally accessible through natural language (NL) brochures, or NL Web pages, which describe the products to potential customers. Building a feature model that hierarchically summarises the different features available in competing products can bring relevant benefits in market analysis. A company can easily visualise existing features, and reason about aspects that are not covered by the available solutions. However, designing a feature model starting from publicly available documents of existing products is a time consuming and error-prone task. In this paper, we present two tools, namely Commonality Mining Tool (CMT) and Feature Diagram Editor (FDE), which can jointly support the feature model definition process. CMT allows mining common and variant features from NL descriptions of existing products, by leveraging a natural language processing (NLP) approach based on *contrastive analysis*, which allows identifying domain-relevant terms from NL documents. FDE takes the commonalities and variabilities extracted by CMT, and renders them in a visual form. Moreover, FDE allows the graphical design and refinement of the final feature model, by means of an intuitive GUI.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements Specification—*analysis,methodologies,specification*

Keywords

Software Product Lines, Variability Mining, Tools

1. INTRODUCTION

When a company enters an established technological market, it shall evaluate the alternative solutions already available, to avoid the development of products that are not novel enough to be appealing, and to reason on possible product features that could bring a competitive advantage in the market. In [12] a set of publicly available documents (*brochures*) has been used to derive a global model, from which specific product requirements for novel systems belonging to the same product line have been derived. The goal of the model was to support the analysis of available Communications-based Train Control Systems (CBTC) products, which are integrated platforms to control the movement of trains within a station and across different stations. The model was represented in the form of a *feature diagram* [15], following the principles of the product line engineering technology. The bottleneck found in the experience was the large amount of human inspection required to identify the common components, as well as the architectural differences, between the solutions proposed by the different vendors. The identification of these *commonalities* and *variabilities* has enabled the definition of mandatory and variant features in the global feature diagram. In order to reduce the time required to extract commonalities and variabilities from the brochures of the different vendors, in [11] we suggested to adopt an automated Natural Language Processing (NLP) approach named *contrastive analysis* to identify *domain-specific terms* (single and multi-word) from textual documents [5]. The proposed method takes the brochures of the different vendors as input, and identifies the linguistic expressions in the documents that can be considered as *terms*. In this context, a *term* is defined as a conceptually independent expression. The domain-specific terms that are common among all the brochures are considered as *commonality candidates*. On the other hand, those domain-specific terms that appear solely in a subset of the brochures are considered as *variability candidates*.

Starting from the experiences in [12] and [11], we have implemented two graphical tools that (1) support the extraction of commonalities and variabilities from natural language (NL) documents, and (2) allow to graphically design a feature model based on the extracted commonalities and variabilities. The first goal is addressed by the Commonality Mining Tool (CMT), while the second goal is addressed by the Feature Diagram Editor (FDE). Though

ACM ISBN X-XXXXX-XX-X/XX/XX.
DOI: http://dx.doi.org/10.1145/0000000.0000000

the definition of the two tools is based on an experience focused on brochures, we advocate that the tools can be used whenever the feature model has to be defined starting from any type of NL documents, including NL requirements. Both the tools are freely available at https://github.com/isti-fmt-nlp/tool-NLPtoFP.

The paper is organised as follows. In Sect. 2, we list the main characteristics of the two tools, and their architecture. In Sect. 3, we describe the NLP approach based on contrastive analysis to identify commonality and variability candidates. In Sect. 4, we describe the details of the two tools. Then, Sect. 5 presents the related work. Finally, Sect. 6 discusses conclusions and future works.

2. OVERVIEW

Commonality Mining Tool (CMT) allows commonalities and variabilities from NL brochures of existing products to be extracted. The main functionalities of CMT are:

1. **Terminology Extraction:** given a set of documents belonging to different vendors, the tool allows the automatic extraction of the domain-specific terms, namely the specific words related to the domain of the product, from each document;

2. **Commonality Candidates Extraction:** the tool automatically identifies of the commonality candidates among the domain-specific terms. These are the domain-specific terms appearing in *all* the documents;

3. **Variability Candidates Extraction:** the tool automatically identifies the variability candidates among the domain-specific terms. These are the domain-specific terms that appear only in a sub-set of the documents;

4. **Documents Surfing:** the user can verify the quality of the selected candidates, by searching the occurrences of candidates in the original documents through the Graphical User Interface (GUI) of CMT;

5. **Commonality/Variability Selection:** among the candidates, the user can select the commonalities and variabilities for the construction of a feature model, manually adding others if needed.

Feature Diagram Editor (FDE) is a tool to define a feature model through the construction of its graphic representation, namely the feature diagram. The main functionalities of FDE are:

1. **Feature Diagram Generation:** the tool automatically defines an initial feature diagram with a set of features selected by the user, based on the commonalities and variabilities produced by CMT;

2. **Feature Diagram Editing:** the user can create, edit and save a feature diagram through a graphical interface based on Drag&Drop operations.

3. **Feature Diagram to Documents Surfing:** the user is guided in surfing the input documents – the same used by CMT – to search for occurrences of features;

4. **SPLOT Import:** the user can import the description of a feature model from the XML format generated by the online tool SPLOT[1] [16] (*.sxfm format). The feature model is automatically rendered in a feature diagram;

5. **SPLOT Export:** the user can export the feature model in the SPLOT format and in *.png image format.

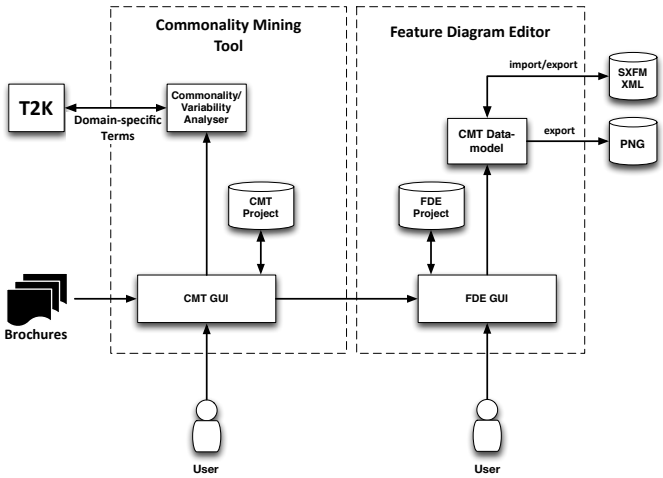

Figure 1: Architecture and interactions of CMT and FDE

The architecture of the two tools and their interaction is shown in Fig. 1. The user interacts with CMT through an intuitive GUI (**CMT GUI**). From the GUI, the user can load the natural language **Brochures** of different vendors and can perform terminology extraction, commonality/variability candidates extraction, document surfing and commonality/variability selection.

The internal engine of CMT (**Commonality/Variability Analyser**) interacts with an external tool named **T2K** [9]. The tool is in charge of performing the terminology extraction, and other NL analysis of the text included in the brochures. CMT allows to store the analysis in a **CMT Project**, which can be saved and loaded by the user.

From CMT, the user can launch FDE. In this case, FDE takes as input the commonalities and variabilities extracted by CMT and stored in the CMT Project. Moreover, a textual version of the original documents is also passed to FDE. The user can interact with the GUI of the tool (**FDE GUI**) to edit the diagram, surf the documents from the features represented in the diagram, or import/export the feature model in the SPLOT format (**SXFM XML**). Moreover, the user can save and load a feature diagram in a **FDE Project**, which includes an XML version of the diagram. FDE can also be executed by the user as a standalone application. In this case, an empty FDE Project is created and the user can start editing the diagram from scratch.

3. THE NLP APPROACH

The method employed by CMT, and supported by T2K [9], is based on a novel natural language processing approach,

[1]http://www.splot-research.org

named *contrastive analysis* [5], for the extraction of *domain-specific terms* from natural language documents. In this context, a *term* is a conceptually independent linguistic unit, which can be composed by a single word or by multiple words. For example, "Automatic Train Protection" is a term, while "Protection" is not a term, since in the textual documents considered in the study reported in [11] it often appears coupled with the same words (i.e., "train", "mission"), and therefore it cannot be considered as conceptually independent.

The *contrastive analysis* technology aims at detecting those terms in a document that are *specific* for the domain of the document under consideration [5, 8]. Roughly, contrastive analysis considers the terms extracted from domain-generic documents (e.g., newspapers), and the terms extracted from the domain-specific document to be analysed. If a term in the domain-specific document highly occurs also in the domain-generic documents, such a term is considered as domain-generic. On the other hand, if the term is not frequent in the domain-generic documents, the term is considered as domain-specific.

In our work, the documents from which we want to extract domain-specific terms are the brochures of different vendors. A brochure is promotional document that describes the product to possible customers. Here, the reasonable assumption is that both commonalities and variabilities can be found among the domain-specific terms of the brochures. The proposed method is summarized in Fig.2. First, conceptually independent expressions (i.e., *terms*) are identified (**Identification of Terms**). Then, **Contrastive Analysis** is applied to select the terms that are domain-specific. From these terms, commonality and variability candidates are extracted (**Commonality/Variability Candidates Identification**). In the tools presented in this paper, the former task is supported by T2K, while the second task is in supported by the Commonality/Variability Analyser component of CMT.

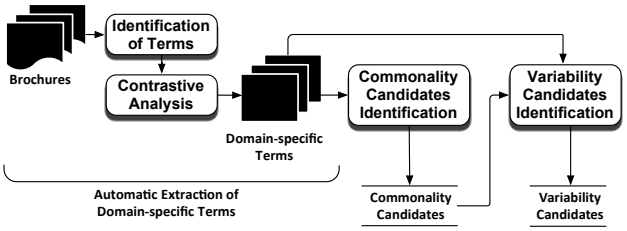

Figure 2: Overview of the approach

3.1 Identification of Terms

Each vendor might have more than one brochure. We collect the brochures of the same vendor i in a single document D_i. Therefore, given n vendors, we have $D_1 \dots D_n$ documents. From each one of these documents we identify a ranked list of *terms*. To this end, we perform the following steps.

POS Tagging: first, Part of Speech (POS) Tagging is performed with an english version of the tool described in [8]. With POS Tagging, each word is associated with its grammatical category (*noun*, *verb*, *adjective*, etc.).

Linguistic Filters: after POS tagging, we select all those words or groups of words (referred in the following as *multi-words*) that follow a set of specific POS patterns (i.e., sequences of POS), that we consider relevant in our context. For example, we will not be interested in those multi-words that end with a preposition, while we are interested in multi-words with a format like <*adjective*, *noun*, *noun*> (such as "Automatic Train Protection").

C-NC Value: terms are finally identified and ranked by computing a "termhood" metric, called C-NC value [5]. This metric establishes how much a word or a multi-word is likely to be conceptually independent from the context in which it appears. The computation of the metric is rather complex, and the explanation of such computation is beyond the scope of this paper. The interested reader can refer to [5] for further details. Here we give an idea of the spirit of the metric. Roughly, a word/multi-word is *conceptually dependent* if it often occurs with the same words (i.e., it is *nested*). Instead a word/multi-word is *conceptually independent* if it occurs in different context (i.e., it is normally accompanied with different words). Hence, a higher C-NC rank is assigned to those words/multi-word that are conceptually independent, while lower values are assigned to words/multi-words that require additional words to be meaningful in the context in which they are uttered.

After this analysis, for each D_i, we have a ranked list of words/multi-words that can be considered *terms*, together with their ranking according to the C-NC metric, and their frequency (i.e., number of occurrences) in D_i. The more a word/multi-word is likely to be a *term*, the higher the ranking. From the list we select the k terms that received the higher ranking. The value of k shall be empirically selected. A higher value guarantees that more domain-specific terms are included in the list. On the other hand, higher values for k might also introduce noisy items, since also words/multi-words with low rank might be included.

3.2 Contrastive Analysis

The previous step leads to a ranked list of k terms where all the terms might be *domain-generic* or *domain-specific*. With the contrastive analysis step, terms are re-ranked according to their domain-specificity. To this end, the proposed approach takes as input: 1) the ranked list of terms extracted from the document D_i; 2) a second list of terms extracted with the same method described in Sect. 3.1 from a set of documents that we will name the *contrastive corpora*. The contrastive corpora is a set of documents containing domain-generic terminology. In particular, we have considered the Penn Treebank corpus, which collects articles from the Wall Street Journal. The reasonable assumption here is that a term that frequently occurs in the Wall Street Journal is not likely to be a domain-specific term of the metro domain. The new rank $R_i(t)$ for a term t extracted from a document D_i is computed according to the function:

$$R_i(t) = \arctan(\log(f_i(t)) \cdot (\frac{f_i(t) \cdot N_c}{F_c(t)})$$

where $f_i(t)$ is the frequency of the term t extracted from D_i, $F_c(t)$ is the sum of the frequencies of t in the contrastive corpora, and N_c is the sum of the frequencies of all the terms extracted from D_i in the contrastive corpora. Roughly, if a term is less frequent in the contrastive corpora, it is considered as a *domain-specific term*, and it is ranked higher. If two terms are equally frequent in the contrastive corpora, but one of them is more frequent in D_i, it is considered as

a term that characterizes the domain more than the other, and, again, it is ranked higher.

After this analysis, for each D_i, we have a list of terms, together with their ranking according the function R, and their frequency in D_i. The more a term is likely to be domain-specific, the higher the ranking. From each list, we select the l terms that received the higher ranking. The choice of l shall be performed empirically: higer values of l tend to include terms that are not domain-specific, while lower values tend to exclude terms that might be relevant in the subsequent phases.

3.3 Commonality Candidates Identification

The commonality candidates are the domain-specific terms that are common to all the documents. Indeed, if a term is *domain-specific* and appears in all the documents of the different vendors, it is likely to be a common feature of all the products. More formally, if $C_1 \dots C_n$ are the sets of domain-specific terms for $D_1 \dots D_n$ respectively, then the set of commonality candidates is defined as: $C = \{C_1 \cap C_2 .... \cap C_n\}$. Ranking is provided also for the set of commonality candidates. The ranking value is provided by computing the average rank of each term.

3.4 Variability Candidates Identification

The variability candidates are identified as those terms which are domain-specific, and therefore appear in some of the C_i sets, but are not part of the commonalities. We assume that, if a domain-specific term appears in some of the documents of the different vendors, but not in all of them, it is likely to be a variant feature, characterizing only a sub-set of the products. More formally, we define the variability candidates as $V = \{C_1 \cup C_2 \dots \cup C_n\} \setminus C$. Also in this case, the ranking value is provided by computing the average rank of each term.

The sets C and V are domain-specific terms of the documents. In order to assess that they actually include commonalities or variabilities, a human operator shall assess the actual relevance of each candidate.

4. CMT AND FDE

In this section we describe the functionalities of the Commonality Mining Tool (CMT) and of the Feature Diagram Editor (FDE). The former employs the approach explained in the previous section to extract commonality and variability candidates, with the support of the tool T2K for domain-specific term extraction (also referred as "terminology extraction" in the following). The latter is used to build a feature diagram.

4.1 Commonality Mining Tool

The Commonality Mining Tool (CMT) provides the extraction of feature candidates starting from the information contained in NL documents that describe similar products. Moreover, among the feature candidates, the tool extracts common and variant feature candidates, to be later evaluated by a human operator (referred in the following as the user).

The idea is to start from a set of NL documents, in pdf/txt format, and extract the set of domain specific terms from these documents. To this end, CMT relies on **T2K (Text-To-Knowledge)** tool [9], which is specifically targeted to identify domain-specific terms.

Once fed with NL documents as input, T2K will provide a set of files containing:

- the NL documents, in txt format;
- the separation into sentences;
- the terminology extraction (i.e., the list of domain-specific terms ranked by relevance);
- the annotation of the text according to the grammar analysis (POS Tagging).

These files will be used to extract the commonality candidates (i.e. domain-specific terms that appear in each document) and variability candidates (all other domain-specific terms). Moreover, the separation into sentences, and the documents in *.txt format will be used to support the identification of relations among the different domain-specific terms extracted.

4.1.1 How CMT Works

A screen-shot of the visual interface provided by CMT is shown in Fig. 3. The internal process followed by CMT can be summarised in the following phases.

Project Set-up.

In this phase, the user creates a CMT Project and loads the NL brochures in *.txt/*.pdf format. The tool assumes that for each vendor, a single document is loaded. Therefore, the user is in charge of merging the different NL documents (through copy/paste or supported by external tools) into a single document. The tool will create a folder for each vendor, which will be used to store the different analysis performed later on.

Terminology Extraction.

In this phase all the NL documents are given as input to CMT, each of them associated to a different folder. For each folder the tool reads the domain-specific terms as they have been processed by T2K. Then, it identifies and stores the position of these terms in the source document(s), and stores the separation into sentences, to be used in the *Color by Cluster* phase of the process described in the following paragraphs.

Extraction of Candidates.

This phase provides the extraction of commonality and variability candidates as follows. Let $D_1 \dots D_n$ is the set of NL documents and T_i the set of relevant terms extracted from the document D_i.

Commonality candidates are computed as:

$$Commonality\ Candidates = \bigcap_{i=1}^{n} Ti$$

Variability candidates are computed as:

$$Variability\ Candidates = \bigcup_{i=1}^{n} Ti - \bigcap_{i=1}^{n} Ti$$

Color by Cluster.

In this phase, colors are assigned to the feature candidates (commonalities and variabilities) to ease the job of the user

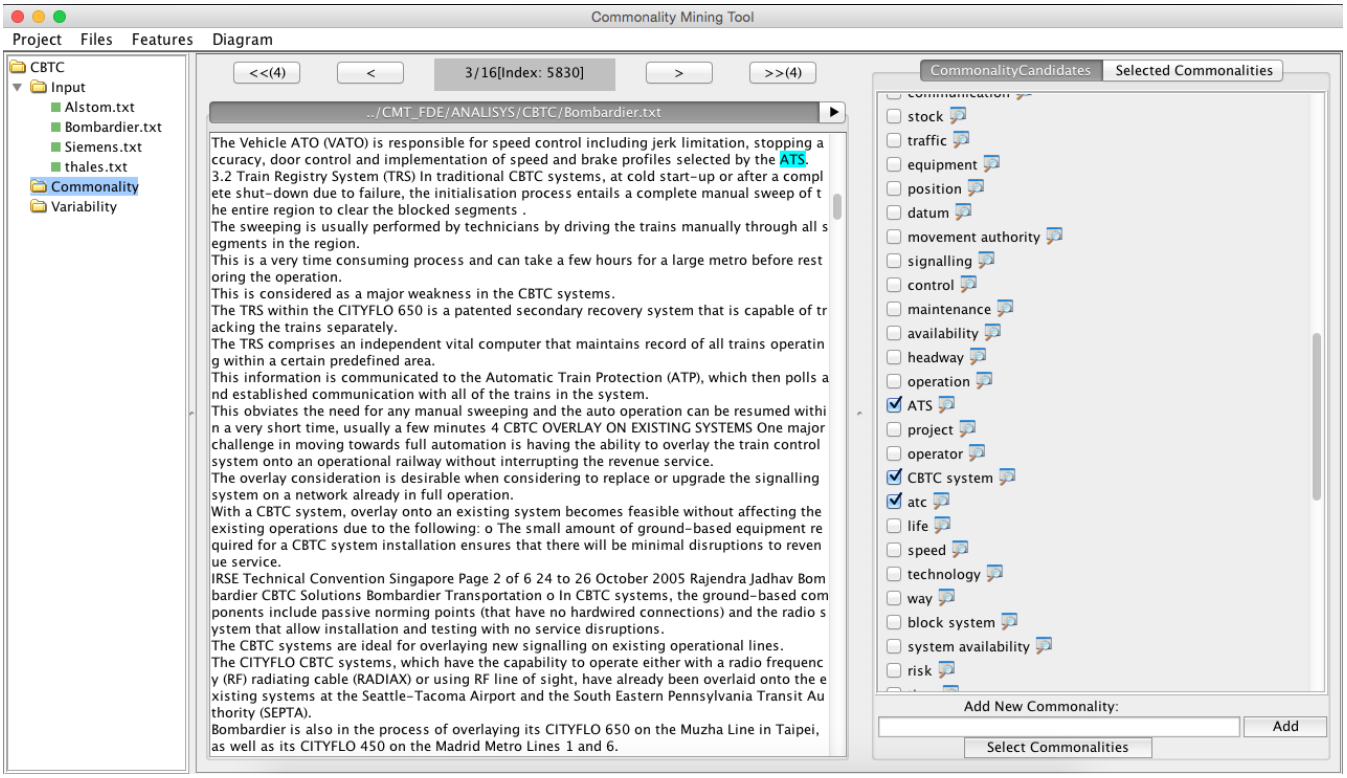

Figure 3: Commonality Mining Tool - The user can surf the original documents to check occurrences of the features in the text. The checked candidates (right panel) are the features that will be passed to FDE.

in understanding the relations among the different features, when such features will be visually shown in FDE. The idea is to assign the same color to *variabilities* that have a textual relation in the original documents. Features associated to domain-specific terms that occur in neighbouring sentences are considered to have a textual relation. Instead, all *commonalities* will be associated to the same color (black, in the default configuration).

To assign colors that highlight relations among variabilities, the position of all the domain specific terms are identified in all the input documents. Such occurrences are used to group the terms in a fixed number of clusters. A cluster identifies a set of terms that have a relation. Here, we use the generic term "relation", without specifying the type of relation, since the relations that we highlight are based solely on the distance of terms within the text. The user will be then in charge of establishing the actual type of relation that occurs among the colored terms: such relation can be a hierarchical one (parent/child feature), a AND/OR relation, or a constraint such as exclude or require. Moreover, such relation can also not exist, since the color highlights relations based on distance in the text, which could not match with semantics relations in the final feature model.

The clustering algorithm adopted to assign colors to clusters is loosely based on K-Nearest Neighbours [20]. A color identifier is assigned to each cluster, which will be associated to all of its terms. The colors will be used by FDE to visualise features that belong to the same cluster. Without going into the details of the algorithm, the reader should imagine that, if two terms are frequently occurring in sentences that are close one to the other, then the terms will be associated to the same color.

The colors associated to each term can be visualised by the user through CMT, but the user will be able to modify the different colors assigned by the algorithm only through FDE. The coloring feature shall be regarded as a *recommendation* of the tool-suite to the user, who will be free to change colors and add new colored features in FDE. Within this work-flow, we do not enforce strict consistency between the colors of the final feature model, and the colors originally generated. Indeed, the goal here is just to *suggest* relations among features in the text, and not to constrain the activity of the user in designing the feature model.

Feature Selection.

During this phase, the user visualises the commonality and variability candidates, checks their occurrences in the input documents, and selects those that seem to be appropriate for the construction of the feature diagram. The user can also manually add other features that s/he thinks necessary. In this phase the user can surf the original documents, by searching the occurrence of a candidate within the text. For example, in Fig. 3, the user is looking at one occurrence of the commonality candidate named "ATS" in one of the original documents. The checked candidates in the right panel of the figure are those that the user has selected as actual commonalities that will be sent to FDE. When the user presses the "Select Commonalities" button at the bottom-right of Fig. 3, the checked candidates becomes visible in the "Selected Commonalities" tab (activated by clicking on the top-right button of Fig. 3). Similar panels and approaches

Figure 4: Feature diagram notations

are provided for variability candidates.

CMT allows searching only one term at a time, and one occurrence of term at a time, to enable accurate inspection of the documents. The search of term occurrences is designed to "remember" the last searched term. In this way, the user can return to such term if, after other searches, there is the need to consider again that term. This functionality is important for the usability of the tool, in order to help discarding the unnecessary terms, and to enable reasoning on the extracted terms by looking at their textual context.

Diagram Generation.

Now the user can run FDE to begin the construction of the feature diagram. If launched by CMT, the commonalities and variabilities selected by the user will be passed to FDE, together with their colors – as assigned by CMT – and their positions and occurrences in the input documents. The tool FDE builds an initial diagram with a root with the same name of the project created with CMT. The selected features are shown as children of such root.

4.2 Feature Diagram Editor

The Feature Diagram Editor aims to define a feature model through the construction of its graphic representation, namely the feature diagram. A feature model is as a hierarchical set of features, and relationships among features. A formal semantics is defined for these models, and each feature model can be characterized by a propositional logic formula [4, 19].

Relationships between a parent feature and its child features (or subfeatures) are categorized as: *AND* - all subfeatures must be selected; *alternative* - only one subfeature can be selected; *OR* - one or more can be selected; *mandatory* - features that are required; *optional* - features that are optional; *a require b*, if the presence of *a* requires the presence of *b* ; *a exclude b*, if the presence of *a* excludes the presence of *b* and vice-versa.

A *feature diagram* is a graphical representation of a feature model [15]. It is a tree where primitive features are leaves and compound features are internal nodes. Common graphical notations are depicted in Fig. 4. These notations are also used by FDE.

A user can start interacting with FDE according to three workflow starting points:

- **from CMT**: in this case the selected features will be given in input, together with their colors and the information about their position in the original texts;
- **as a standalone application**: in this case, the user can edit the diagram from scratch without relying on previously extracted features;
- **importing an SXFM files**: an SXFM file is an .xml file generated with the tool SPLOT [16]. In this case, FDE will automatically generate the Feature Diagram corresponding to the feature model defined in such file.

Basic Operations.

FDE is used mainly by means of Drag&Drop operations. Fig. 5 shows the interface of the tool[2] (ignore at this stage the "Search Feature" label in the figure). FDE has a palette on the left with the graphical symbols already reported in Fig. 4 (AND decomposition can be performed by combining the mandatory/optional connectors). The user can select one of the symbols from the palette and drag it to the central dashboard, to build or update the feature diagram. With this user-friendly approach, new features can be introduced, as well as connections among features.

Some functionalities of FDE are activated by means of a pop-up menu that can be opened by right clicking on a feature. Among them, the change of the name of the feature, or the opening of a window to search occurrences of the feature in the original documents.

Finally, saving, loading, import and export operations can be accessed through the menu bar of the tool (under the "Files" menu). Here, it is worth noting that, when saved, the visual diagram is mapped to a formal model expressed in XML. When exported in the *.sxfm format, such model can also be read by the SPLOT tool [16], which allows performing additional analysis on the product family associated to the model.

Surfing the Documents.

The workflow of FDE highly depends on the user preferences and needs. However, here it is useful to describe how the user can surf the original documents of the different vendors starting from the visual representation of the feature diagram.

As shown in Fig. 5, the user can select a group of features, and right click to search them in the original texts. In Fig. 5, the user has selected a group of two features, named "CBTC System" and "ATS" (a component of the CBTC system). When the user presses "Search Feature", FDE opens the window shown in Fig. 4.2. From such window the user can see the occurrences of the selected features in the original documents.

It is worth noting that the colors displayed in this window have a different meaning with respect to those generated by CMT, and shown in the feature diagram. Here, the colors serve to understand whether the feature was extracted from the text as a commonality, a variability or was an additional feature not previously extracted from the text, as shown in the legend at the top-left of Fig. 4.2.

4.3 Tool Download

CMT and FDE have been developed in Java, to ensure their portability. The source code can be freely downloaded from https://github.com/isti-fmt-nlp/tool-NLPtoFP, together with some illustrative examples.

After downloading the tools, which are embedded in a single project, the user can import them as a Maven project[3] within the Eclipse[4] platform. Both tools are under LGPL license. FDE can be executed as-is. Instead, terminology extraction through CMT is performed remotely. Interested

[2]The colors of the feature diagram in the figure have been adjusted by the user. Indeed, right after importing the features from CMT, all the commonalities are normally colored in black.

[3]https://maven.apache.org

[4]http://www.eclipse.org

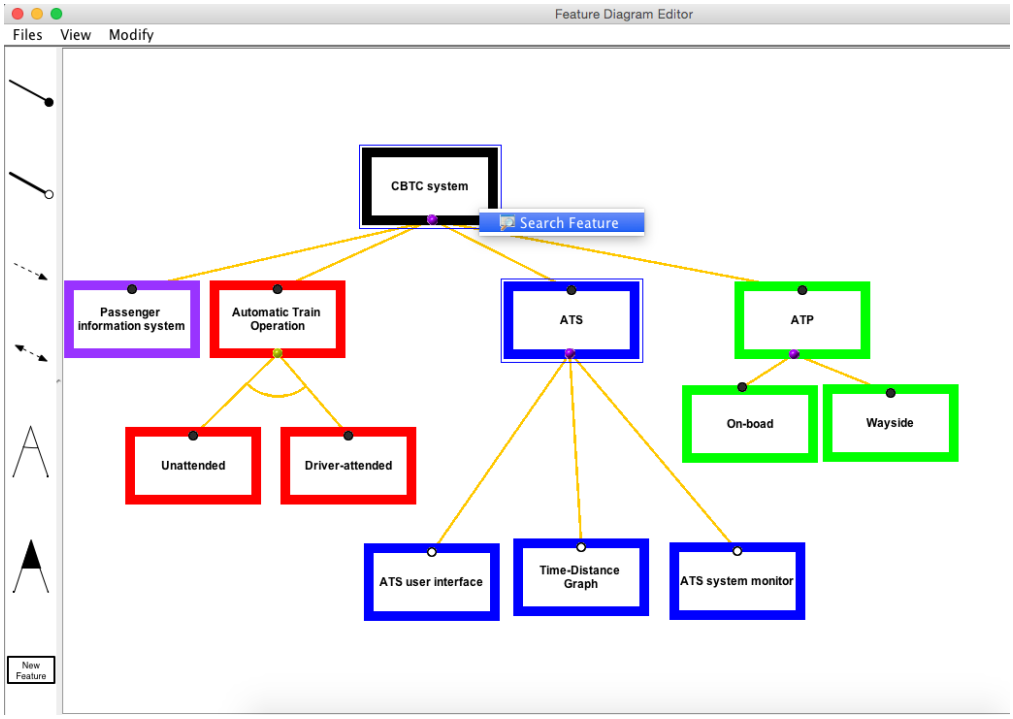

Figure 5: Feature Diagram Editor - The tool allows building a feature diagram through Drag & Drop operations, by using the palette on the left and dragging the graphical elements to the central dashboard.

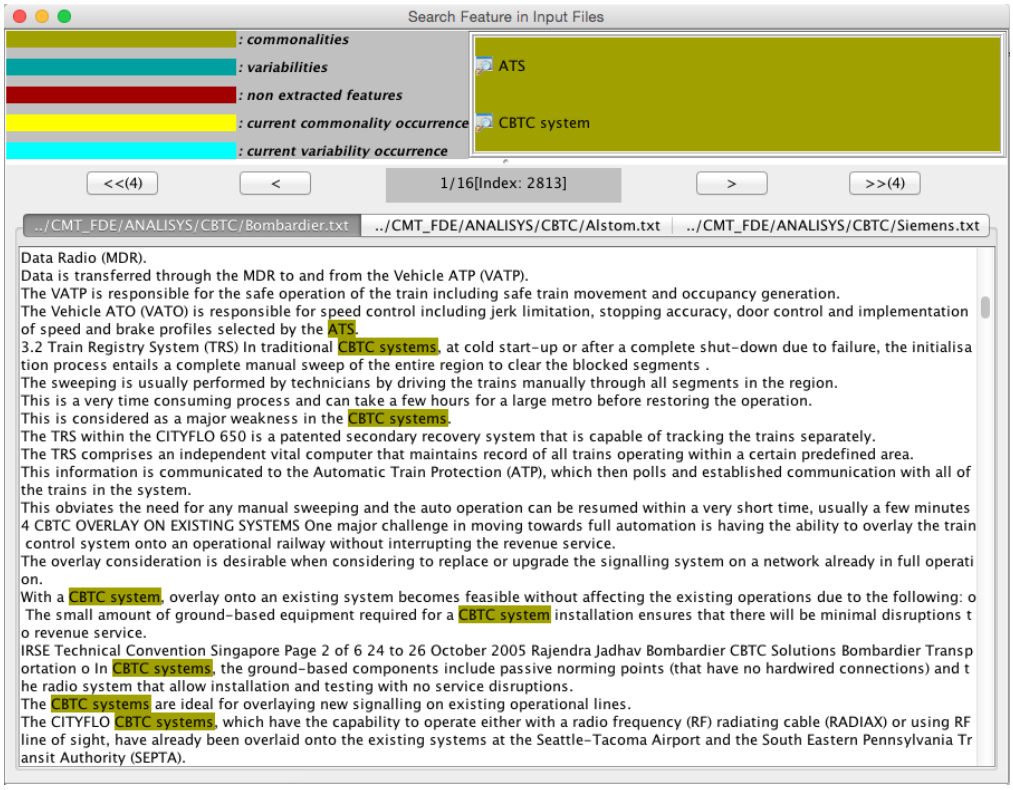

Figure 6: Feature Diagram Editor - The tool allows to inspect the original documents, according to the features selected in the feature diagram.

users shall contact the authors of the current paper to get an account that will allow them to perform the terminology extraction task.

5. RELATED WORK

Mining commonalities and variabilities from natural language documents is an open issue in product line engineering, with several solutions proposed in the literature [1]. In general, the approaches are based on two steps: *feature mining* and *feature model synthesis*. The first step aims at identifying features from documents, while the second step is oriented to automatically building the feature model. Since in this paper we focus on feature mining, we compare the works according to the methodology applied to identify features.

Most of the works focus on the extraction of features from natural language requirements and legacy documentation [13, 6, 3, 17, 18, 21]. The DARE tool [13] is the earliest contribution in this sense. A semi-automated approach is employed to identify features according to lexical analysis based on term frequency (i.e., frequently used terms are considered more relevant for the domain). Chen *et al.* [6] suggests the usage of the *clustering* technology to identify features: requirements are grouped together according to their similarity, and each group of requirements represents a feature. Clustering is also employed in the subsequent works [3, 17, 18, 21], but while in [6] the computation of the similarity among requirements is *manual*, in the other works *automated* approaches are employed. In particular, [3] uses IR-based methods, namely the Vector Similarity Metric (VSM) and Latent Semantic Analysis (LSA). With VSM, requirements are represented as vectors of terms, and compared by computing the cosine among the vectors. With LSA, requirements are similar if they contain semantically similar terms. Two terms are considered semantically similar if they normally occur together in the requirements document. LSA is also employed by Weston *et al.* [21], aided with syntactic and semantic analysis, to extract the so-called Early Aspects. These are cross-cutting concerns that are useful to derive features. Niu *et al.* [17, 18] use Lexical Affinities (LA) – roughly, term co-occurrences – as the basis to find representative expressions (named Functional Requirements Profiles) in functional requirements.

All the previously cited works use *requirements* as the main source for feature mining. Other works [14, 10, 2, 7] present approaches where *public product descriptions* are employed, like in our case. While in [14] the feature extraction process is manual, the other papers suggest automated approaches. The feature mining methodology presented in [10] is based on clustering, and the authors provide also automated approaches for recommending useful features for new products. Instead, the approach presented in [2] is based on searching for variability patterns within tables where the description of the products are stored in a semi-structured manner. Finally, the approach in [7] uses text similarity measures to support the clustering of different terms into features. Both [2] and [7] include also a relevant part of feature model synthesis.

For a recent literature review of the related work in feature mining from NL documents, the interested reader can refer to Bakar *et al.* [1].

Regardless of the technology, the main difference between [10], [2] and our work is that the former two rely on feature descriptions that are rather structured. Indeed, in [10] the features of a product are expressed with short sentences in a bullet-list form, while in [2] features are stored in a tabular format. Instead, in our case we deal with brochures with less structured text, where the features have to be discovered *within* the sentences. The feature mining approach employed in [7] mainly differs from ours in the natural language processing technologies adopted. In such paper, information retrieval and clustering techniques are used to mine features, while here we propose the usage of the novel contrastive analysis technology for the extraction of domain-specific terms. Moreover, in our paper we also provide intuitive, user-friendly GUIs, which are not provided by any of the previous works.

The novelties of the current work w.r.t. the other papers are: 1) the usage of free-text informative brochures as the input documents for the commonality/variability mining process; 2) the usage of contrastive analysis for the extraction of domain-specific terms; 3) the introduction of a user-friendly tool for commonality/variability mining (CMT); 4) the introduction of a user-friendly tool for editing feature diagrams (FDE).

6. CONCLUSION

In this paper, we have presented two tools, namely CMT and FDE, which can ease domain analysis when a company wishes to enter a new market. The two tools are both in a prototypical academic version, and several improvements are still needed to make them industrially applicable. Besides the look-and-feel improvements that are required, we plan to extend FDE with the introduction of minimum and maximum cardinalities in features and group of features. Moreover, we also plan to experiment the usage of the tools in real-world scenarios, to monitor how a user builds a feature model starting from NL documents. In our view, this user-based observation is fundamental to understand how to introduce feature-model synthesis approaches (as, e.g., in Davril *et al.* [7]) in a CMT/FDE-based tool-chain.

We also advocate the usage of the proposed tools – in particular CMT – for mining common and variant features form NL requirements, and not only from informal product descriptions. In principle, requirements documents of similar products can be regarded as the brochures of different vendors, and processed according to the approach defined in this paper. In this case, the final output would be a feature diagram – and a corresponding feature model – that represents the product line associated to the requirements.

7. ACKNOWLEDGMENTS

This work was partially supported by the LearnPAd FP7-ICT-2013.8.2 European Project.

8. REFERENCES

[1] Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software*, (0):–, 2015. Available online 9 May 2015.

[2] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire. On extracting feature models from product descriptions. In *Proc. of VaMoS '12*, pages 45–54, 2012.

[3] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler. An exploratory study of information retrieval techniques in domain analysis. In *Proc. of SPLC '08*, pages 67–76, 2008.

[4] D. S. Batory. Feature models, grammars, and propositional formulas. In *Proc. of SPLC*, pages 7–20, 2005.

[5] F. Bonin, F. Dell'Orletta, S. Montemagni, and G. Venturi. A contrastive approach to multi-word extraction from domain-specific corpora. In *Proc. of LREC'10*, pages 19–21, 2010.

[6] K. Chen, W. Zhang, H. Zhao, and H. Mei. An approach to constructing feature models based on requirements clustering. In *Proc. of RE'05*, pages 31 – 40, 2005.

[7] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 290–300. ACM, 2013.

[8] F. Dell'Orletta. Ensemble system for part-of-speech tagging. In *Proc. of Evalita'09, Evaluation of NLP and Speech Tools for Italian*, 2009.

[9] F. Dell'Orletta, G. Venturi, A. Cimino, and S. Montemagni. T2kˆ2: a system for automatically extracting and organizing knowledge from texts. In N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*, pages 2062–2070. European Language Resources Association (ELRA), 2014.

[10] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *Proc. of ICSE'11*, pages 181–190, 2011.

[11] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta. Mining commonalities and variabilities from natural language documents. In *17th International Software Product Line Conference, SPLC 2013, Tokyo, Japan - August 26 - 30, 2013*, pages 116–120, 2013.

[12] A. Ferrari, G. O. Spagnolo, G. Martelli, and S. Menabeni. Product Line Engineering Applied to CBTC Systems Development. In *Proc. of ISOLA'12*, volume 7610 of *LNCS*, pages 216–230, 2012.

[13] W. Frakes, R. Prieto-Diaz, and C. Fox. Dare: Domain analysis and reuse environment. *Ann. Softw. Eng.*, 5:125–141, Jan. 1998.

[14] I. John. Capturing product line information from legacy user documentation. In *Software Product Lines*, pages 127–159. Springer, 2006.

[15] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical report, Carnegie-Mellon University Software Engineering Institute, 1990.

[16] M. Mendonca, M. Branco, and D. Cowan. S.p.l.o.t.: Software product lines online tools. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, pages 761–762. ACM, 2009.

[17] N. Niu and S. M. Easterbrook. Extracting and modeling product line functional requirements. In *Proc. of RE'08*, pages 155–164, 2008.

[18] N. Niu and S. M. Easterbrook. On-demand cluster analysis for product line functional requirements. In *Proc. of SPLC'08*, pages 87–96, 2008.

[19] F. Roos-Frantz. *Automated Analysis of Software Product Lines with Orthogonal Variability Models: Extending the FaMa Ecosystem*. PhD thesis, University of Seville, 2012.

[20] S. Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667–671, 2005.

[21] N. Weston, R. Chitchyan, and A. Rashid. A framework for constructing semantically composable feature models from natural language requirements. In *Proc. of SPLC '09*, pages 211–220, 2009.