From Generic Requirements to Variability

Alessandro Fantechi
Dipartimento di Ingegneria dell'Informazione, Università di Firenze, Italy
alessandro.fantechi@unifi.it
Stefania Gnesi
Istituto di Scienza e Tecnologie dell'Informazione "A.Faedo"
Consiglio Nazionale delle Ricerche, ISTI-CNR, Pisa, Italy
stefania.gnesi@isti.cnr.it
Laura Semini
Dipartimento di Informatica, Università di Pisa, Italy
laura.semini@unipi.it

Abstract

This paper describes a research activity aiming at extracting variability information from ambiguities and vagueness of generic requirement documents, written in Natural Language. The proposed activity continues a research stream focusing on techniques to extract variability information from requirement documents. Here, we study the introduction of a process able to distinguish structural from functional variability, both in the extracted variability model and in the derived lower-level requirements. The problem is stated with reference to an example, a solution proposal is sketched together with related research questions, and a validation path is envisaged.

1 Introduction

Software product line engineering (SPLE) aims at developing a line of products using a shared platform or architecture (commonalities) and mass customization (variabilities), with the goal of maximizing the commonalities whilst minimizing the cost of variations (i.e., of individual products), thus specifically facilitating reuse in a predictive manner. A variation point identifies a set of possible variants, a variant represents a specific product feature.

Among the fundamental activities of software product line engineering (SPLE) there is the identification of the variability in different artifacts of the system, such as requirements, architecture and test cases [CABA09]. Several methods and tools were developed for variability identification and management, that are specifically focused on *requirements*, including feature-oriented domain analysis (FODA) [KCH+90], the RequiLine tool [vdML03], the domain requirements model (DRM) based approach [PKS04], as well as the work of Moon *et al.* [MYC05].

In previous works [FGS17, FFGS18a, FFGS18b], we have discussed the potential that Natural Language (NL) requirements documents exhibit as a source to identify variability information. This information can be later used to define variability models from which different systems can be generated. In recent years the increasing capabilities of Natural Language Processing (NLP) tools [Got16] has driven the search for variability identification methods based on extracting features and variability-related information from NL documents [NBA+17, FSD13, LSS17, IRBW16, BKS15]. In particular, in the requirements engineering of software product lines (SPL), several researches have focused on exploiting natural language processing (NLP) techniques and tools to extract information related to features and variability from requirement documents.

Our previous work in this area has focused on the most effective way of extracting potential variability in requirements, looking for the most suitable textual indicators and for the analysis tools able to detect such indicators. Following the common practice in SPLE, we have hence modeled variability using features and feature diagrams [KCH+90]. We considered real detailed requirement documents coming both from industry and academia and the purpose of the research was to detect the ambiguities still present in these documents, intrinsic in natural language usage, and to understand if these could hide some variability.

In this paper we refine this approach by considering instead *high level/generic* requirements documents, in which requirements are given at a high abstraction level, like the declarative description of a main functionality, or the very first, coarse grained, physical description of a system. The idea is that high level requirements can hide a family of different products, so that they may be seen as generic requirements of a potential family of systems, and their specialization may correspond to product specific requirements. In this paper, variability extraction is based on the observation that a high level requirement document may describe what a potential family of systems will do, and variability is given by different choices among the implementation alternatives.

Another novelty of the present work is that variability is extracted by separating *functional* and *structural* perspectives by means of different feature diagrams:

- according to the functional perspective, a feature is a function or service offered by the system, and its sub-features represent special cases or different variants of such a functionality;
- according to the structural perspective, the feature diagram defines a domain or an architectural organization.

Specifically, the idea underlying this research activity is to combine:

- our approach to variability identification through the use of a NLP tool;
- the classification of features as functional and structural as introduced in [IR14], that improve features description with respect to the practice of mixing up in a single diagram concepts that would rather be separated;
- the idea that the specialization processes of functional and structural requirements are interwound. Figure 1 illustrates this double specialization in a UML like formalism: on the top line there are an abstract domain element and a generic functional requirement referring to it; on the bottom line there are the refinements.

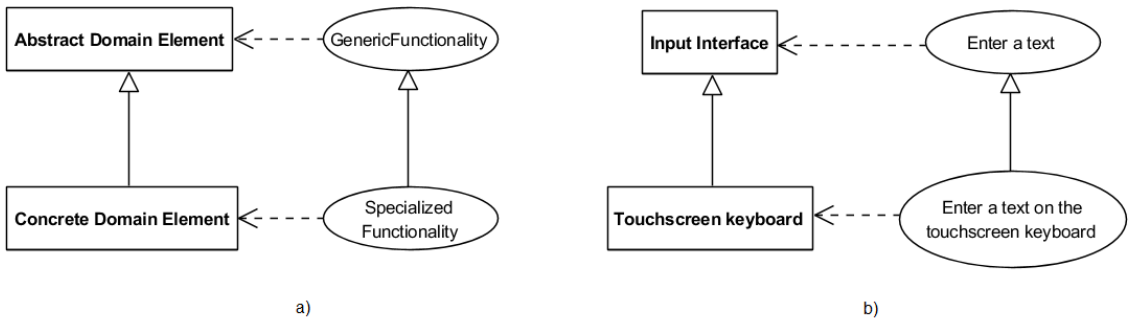

Figure 1: Paired specializations in a refinement: the general case (a) and an example (b).

This paper aims at describing this combined approach, showing the relevant issues and defining a process that, moving from generic requirements documents, builds two feature diagrams expressing variability along the different perspectives. We illustrate the approach on a small case study, leaving the validation effort on requirement documents coming from industrial case studies to further work.

This paper is structured as follows: background for this paper is given in Section 2, including more detailed reference to our previous work. Section 3 describes the proposed approach with a simple example and in more general terms. Section 4 discusses the prospected research activity, formulating specific research questions and envisaging a validation plan, recurring to available industrial case studies.

2 Background

2.1 Variability and Feature Diagrams

Feature models have been extensively used in the area of product line engineering. Product line engineering provides a way to manage variability during the entire design process and is an important means for identifying variability needs early on.
Feature models are visually represented by means of feature diagrams, introduced in [KCH+90] as a graphical formalism, *and/or* hierarchy of features, to describe in a uniform way a variety of different possible implementations of a system. The variability depends on which features are included and which ones are not. The features are represented as the nodes of a tree. Features come in several flavours, (Figure 2 illustrates the graphical constructs):

- **Optional** features may be present in a system only if their parent is present;
- **Mandatory** features are present in a system if and only if their parent is present;
- **Alternative** features are a set of features among which one and only one is present in a system, provided their parent is present.

Constraints may be added to a feature diagram permitting for example to express that the presence of a feature **requires** the presence of another one.

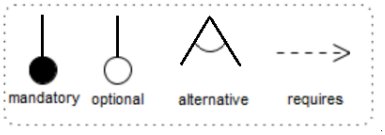

Figure 2: Basic constructs of a feature diagram.

A *Feature model* is hence a compact representation of the commonalities and variabilities of a family of systems, expressed as mandatory, optional features and constraints.

2.2 Extracting variability from ambiguities in NL requirements

In previous works [FGS17, FFGS18a, FFGS18b], we have defined an approach to extract variability issues from requirement documents using Natural Language analysis tools. These tools are aimed at revealing the ambiguity defects of the NL sentences in the requirements document.

In line with this stream of research, in [FGS17] we concentrated on the correspondence between ambiguity indicators and feature diagram fragments, to define a systematic way of building a feature diagram from a set of variability points extracted with a NLP tool. In particular we have focused on using tools aimed at revealing the *ambiguity* defects of the NL sentences in the requirements document. The underlying intuition is that often ambiguity in requirements is due to the (conscious or subconscious) need to postpone choices for later decisions in the implementation of the system. Ambiguity in NL has been largely studied in requirements engineering (RE), and several approaches have been developed to automatically detect defective expressions that can be interpreted in different ways by the stakeholders who have to read the requirements [TB13, GCK10, BK04, RFG+17, FFWE17]. These approaches focus on identifying typically vague terms, such as adjectives and adverbs (e.g., [TB13, GCK10]), and ambiguous syntactic construction due to the use of pronouns [YDRG+10], or coordinating conjunctions such as “and” or “or” [YWDRN10, CNDRW06]. Our work considers ambiguity not as a defect, but as a means to enlighten possible variation points in an early phase of software and system development, and to give space for a range of different products.

Then, in [FFGS18a] we initiated a validation of the approach with the QuARS (Quality Analyser for Requirements Specifications) NLP tool [GLT05], in order to better assess the relation between ambiguity detection by QuARS and potential variation points. The evaluation has shown that some ambiguous terms, such as “and/or” “or”, and weak terms such as “may” or “could” are more likely to indicate variability rather than ambiguity. Instead, typically vague terms, such as “adequate”, “significant”, etc. are more likely to indicate ambiguity. In [FFGS18b] we have analysed the ambiguities returned by the tool on three different sets of requirements, classifying them in false positives, real ambiguities, and variation points.

3 From ambiguities in the domain/structure to specialization and variability of the requirements

The work cited in the previous section has mostly regarded the search for the most effective way of extracting potential variability in requirements, looking at the most suitable indicators, and on the analysis tools able to detect such indicators. Only [FGS17] included the construction of a feature diagram to express the variability revealed by the considered techniques.

To advance our research on the topic, we now focus back on modeling the revealed variability, introducing the distinction between different perspectives (typically, structural and functional perspectives) according to which variability can be interpreted. The idea is that generic requirements can hide a family of different products, that can be revealed looking at different specializations both under a structural and a functional perspective.

3.1 The research idea through a simple example

In order to introduce a systematic variability extraction process based on the above principles, we discuss a simple example. The starting point is a high level, generic, set of requirements. For instance, the following can likely be two high level requirements of a generic mobile phone:

Rg1. The phone shall offer a suitable interface to enter a text.

Rg2. The phone shall echo the inserted text on a screen.

The two requirements describe two generic functionalities that a mobile phone shall offer to the user: entering a text, and seeing the text that she is entering. Observing them from a structural perspective, they say that a mobile phone must have an input interface and a screen, that can be considered as structural, mandatory, features: this is summarized by the feature diagram of Figure 3.a.

Considering them from a functional perspective, they mandate that the system offers two functions, again represented as two mandatory features in Figure 3.b.

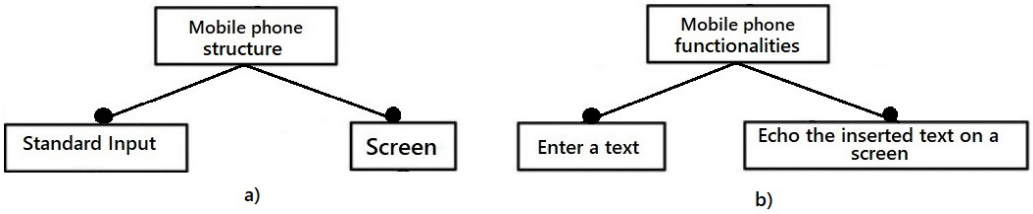

Figure 3: Generic features of a mobile phone: *a.* structural perspective, *b.* functional perspective

When analyzing the requirements Rg1 and Rg2, looking for vague and underspecified terms according to [FFGS18b], *suitable* is detected as a *vague* word that needs to be specialized. In our case, the *suitable interface* is intended to be a structural feature of the system, for which two alternatives could be identified: touchscreen keyboard and microphone.

Moreover, *screen* is also detected as an underspecified term that needs to be instantiated, in this case the analyst decides instantiations to be touchscreen and non-touch screen.

As a first outcome of this analysis effort, we can hence add two requirements, each describing two alternatives:

Rs1. The input modalities in a mobile phone device are the touchscreen or the old style 3x4 physical keyboard.

Rs2. The output modalities in a mobile phone device are the touchscreen or the non-touch screen.

The information that has been added to the description of the mobile phone family induces a refinement of the functional part of requirements **Rg1** and **Rg2** to consider the new structural elements. The analyst decides to offer normal editing and, optionally, also speech recognition:

Rs3. The mobile phone shall permit the user to enter a text through the touchscreen keyboard or through the 3x4 physical keyboard.

Rs4. The mobile phone may (optionally) permit the user to enter a text through voice dictation.

Rs5. The phone shall echo the inserted characters on the touch or non-touch screen as they are inserted.

Rs6. The phone may (optionally) echo the dictated words on the touchscreen as they are recognized.

These requirements introduce different variants of the functional features. They also require to consider the microphone among the standard inputs. Even though speech recognition is optional, we consider the microphone as a mandatory element since it is included in all phones, and refine **Rs1** in **Rs1'**

Rs1'. The input modalities in a mobile phone device are the touchscreen or the old style 3x4 physical keyboard, and the microphone.

In terms of feature diagrams, the structural refinement is described in Figure 4, and the functional variants can be expressed by the feature diagram of Figure 5.

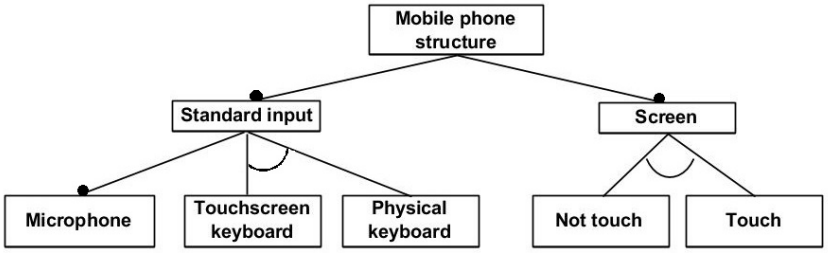

Figure 4: The refined structural features of a mobile phone

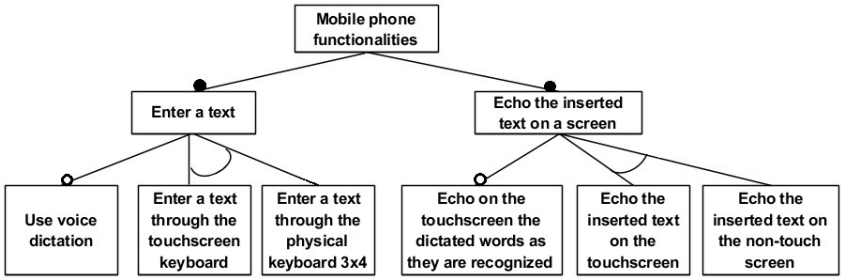

Figure 5: The refined functional features of a mobile phone.

We observe that the features in the diagrams in Fig 4 correspond to those in the diagram in Fig. 5 and could be (almost) pairwise related by inter-diagram **requires** dependencies: each functional feature requires a corresponding structural entity: e.g., enter a text **requires** a standard input; enter a text through the touchscreen keyboard **requires** a touchscreen keyboard, echoing characters or words on the touchscreen **requires** a touchscreen. We document these dependencies in Figure 6, where, to simplify the picture, we consider only a fragment of the each feature diagrams.

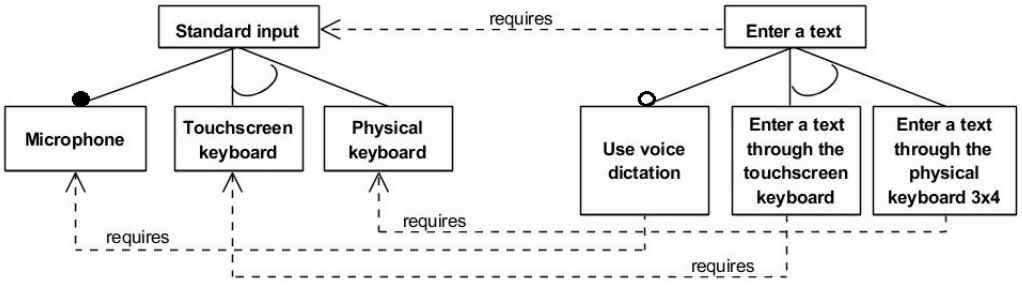

Figure 6: Inter-diagram *requires* dependencies (on a simplified model).

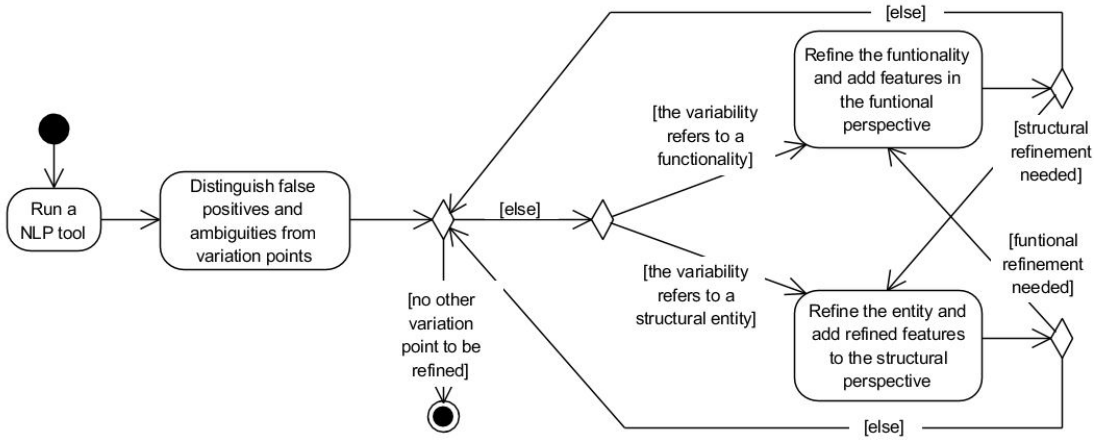

Figure 7: The process to extract variability from generic requirements.

3.2 The research idea in general terms

We propose a requirements refinement process, that begins with a list of (generic) requirements.

A generic requirement consists of key generic functionalities and implementations that can become a differentiating factor for a user. A requirement can be generic because:

1. it describes a functionality, in terms of what the system is supposed to do, and not about how it does so; the functionality is a high level functionality and does not take into account special cases, alternatives, etc. Examples are: *The phone shall permit to* **send a message**, *The phone shall offer the possibility to* **enter a text**.

2. it predicates about a domain element that has still to be better detailed (by inheritance or by composition). As an example: *The phone shall include a* **screen**.

In all these cases, the refinement process can reveal variability points and can lead to lower level requirements that define a family of products that differ in the implementation platform, in the offered functionalities, or in the adopted physical elements (that roughly correspond to different domain elements).

The requirement refinement process can be given a more systematic description as shown in Figure 7 and discussed below:

1. The first step of the process runs a NLP tool to analyze the generic requirements and to look for indicators of under-specifications and of vagueness. We recall from [FFGS18b] that under-specified requirements are detected looking for terms needing to be instantiated (e.g.: *information, interface, access, button, message, screen, ...*), and vague requirements are those including undetermined adjectives and adverbs like: *clear, easy, strong, good, bad, useful, significant, adequate, recent, ...*

   With reference to the mobile phone example, this means to run the tool on Rg1 and Rg2. We used QuARS to this purpose, having *screen* and *suitable* detected as defective words.

2. Once the under-specified or vague requirements have been singled out, their analysis (second step of the process) is guided by the questions: "Is this a false positive?", "Is this just an ambiguity?", or "Can this requirement hide a variation point?". More concretely, both in the case of vagueness and in the case of under-specification, the criterion to identify a variability is the existence of more than one possible instance of the defective term.

   This is a manual step, made by an analyst with some knowledge of the domain. In our running example, both the screen and the *suitable interface* were considered as possible variability points.

3. The third step of the process is to refine the detected variation points:

   (a) In the case of generic and vague functionality, the requirement is specialized by a set of concrete functionalities. For instance the requirement "The phone shall permit to send a message" may be refined

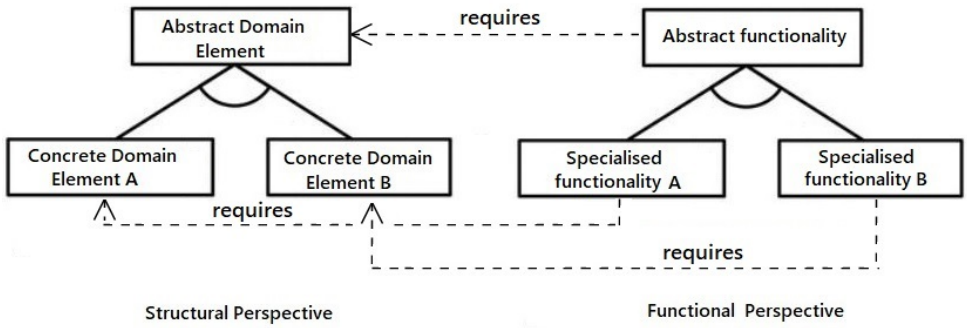

Figure 8: Revealing variability: the difference in perspective (structural/functional) leads to a pair of different (but related) feature diagrams to specialize generic requirements.

in a mandatory feature "send a SMS message" and two optional features: "send a MMS message" and "send a WhatsApp message". The specialization may need the definition of structural/architectural aspects of the system, that may in turn lead to a further refinement of the functionalities if there are still some vaguenesses or under-specifications.

(b) In the case of a requirement containing a defective term that denotes a domain entity we consider all the possible specializations of the generic term and refine the structural description. This leads in almost all the cases to a refinement in the functional perspective, to take into account the newly introduced concrete domain elements.

This is the case of our running example, where different input interfaces have been considered: the keyboard and the microphone, that are both mandatory in a phone, and where the keyboard can be either on the touchscreen or the 3x4 physical keyboard. The considered screens have been the touchscreen and the not touch screen. This refinement has led to Rs1 and Rs2.

In general, the third step of the process is iterative and alternates structural and functional specializations. In the case of the mobile phone, the structural refinement has naturally driven the functional refinement step (Figure 5): after having described the phone structure, it has been clear that the functionality *enter a text* could have been implemented by editing on a touch or on a traditional keyboard (Rs3). The analysis has also singled out a third choice, namely voice dictation (Rs5). Each choice has a corresponding echoing modality (Rs4 and Rs6). Finally, speech recognition has induced a further structural refinement, to consider the microphone (Rs1').

The final result is a pair of feature diagrams, one in functional and the other in structural perspective. These diagrams can be connected through **requires** dependencies, relating a functional feature with the structural entity(-ies) needed to implement it, as exemplified in in Figure 6 and generalized in Figure 8.

4 Conclusion and Discussion

We have presented the current status of a research activity aiming at driving the elicitation of a family of products from an initial set of generic requirements written in Natural Language. The approach is based on extracting variability information from ambiguities and vagueness of the generic requirement documents.

When specializing generic requirements of a (potential) family of systems, variability is typically introduced both on a structural and on a functional perspective, hence there is the need to introduce a process able to distinguish the structural from the functional perspectives in eliciting variability, both in the extracted variability model and in the derived lower-level requirements. We have presented the problem by means of a simple example and we have sketched a solution proposal.

We plan to validate the presented approach by application to industrial case studies. Our research agenda hence consists in taking some real world requirements document, apply a NL processing tool to look for under-specifications and vaguenesses, and perform the analysis and refinement step, to answer to the following research questions:

Research Question 1: Is there a preferred order in the process of variability identification? i.e. is it better to first address refinement of the structural features rather than the functional ones, or viceversa? Or the two parallel processes should actually interleave?

Research Question 2: How well can the approach be accepted in industry? How much industrial users will appreciate this approach in terms of perceived usefulness and perceived ease-of-use (regarding both aspects of using NLP analysis tools and of distinguishing functional and structural models)?

Regarding the first research question, we have already identified an interesting candidate case study in the Clarus Weather System. Clarus is an initiative sponsored by the Federal Highway Administration (FHWA) to organize and make more effective environmental and road condition observation capabilities. The list of requirements of the Clarus system is organised in two documents: *high level* and *detailed* system requirements. These are considered suitable to validate the capabilities of the NLP tools to indicate potential variability: any detected variability should be reflected, either under a structural or a functional perspective, to specialized requirements. We plan to use them as a testbed for our approach applying the refinement process to the generic requirements and comparing the outcome with their detailed requirements. An example of generic requirement is the one telling that "the Clarus system shall employ industrial standards". The under-specified wording *industrial standard* calls for an analysis and refinement effort. Indeed, we found a comment specifying the family of standards of interest, and in the detailed document the defective requirement is substituted by 10 different requirements, one for each concrete standard.

Other industrial case studies will then be needed to be able to generalize the answers to the question.

The second research question will require a longer term collaboration with one or more companies, in order to involve requirement engineers and analysts in an investigation including pilot projects and interviews, within an established technology acceptance model.

References

[BK04] D. M. Berry and E. Kamsties. Ambiguity in requirements specification. In *Perspectives on software requirements*, pages 7–44. Springer, 2004.

[BKS15] N. H. Bakar, Z. M. Kasirun, and N. Salleh. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software*, 106:132–149, 2015.

[CABA09] Lianping Chen, M. A. Babar, and N. Ali. Variability management in software product lines: a systematic review. In *Proc. of the 13th Int. Software Product Line Conference*, pages 81–90. Carnegie Mellon University, 2009.

[CNDRW06] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis. Identifying nocuous ambiguities in natural language requirements. In *Requirements Engineering, 14th IEEE International Conference*, pages 59–68. IEEE, 2006.

[FFGS18a] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini. Hacking an ambiguity detection tool to extract variation points: an experience report. In *Proc. of the 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2018, Madrid, Feb. 2018*, pages 43–50. ACM, 2018.

[FFGS18b] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini. Requirement engineering of software product lines: Extracting variability using NLP. In *26th IEEE Int. Requirements Engineering Conference, RE 2018, Banff, AB, Canada, Aug. 2018*, pages 418–423. IEEE Computer Society.

[FFWE17] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder. Rapid quality assurance with requirements smells. *Journal of Systems and Software*, 123:190–213, 2017.

[FGS17] A. Fantechi, S. Gnesi, and L. Semini. Ambiguity defects as variation points in requirements. In *Proc. of the 11th International Workshop on Variability Modelling of Software-intensive Systems*, VAMOS '17, pages 13–19, 2017. ACM.

[FSD13] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta. Mining commonalities and variabilities from natural language documents. In *Proc. of the 17th International Software Product Line Conference*, pages 116–120. ACM, 2013.

[GCK10] B. Gleich, O. Creighton, and L. Kof. Ambiguity detection: Towards a tool explaining ambiguity sources. *Requirements Engineering: Foundation for Software Quality*, pages 218–232, 2010.

[GLT05] S. Gnesi, G. Lami, and G. Trentanni. An automatic tool for the analysis of natural language requirements. *Comput. Syst. Sci. Eng.*, 20(1), 2005.

[Got16] G. Goth. Deep or shallow, nlp is breaking out. *Communications of the ACM*, 59(3):13–16, 2016.

[IR14] N. Itzik and I. Reinhartz-Berger. Generating feature models from requirements: structural vs. functional perspectives. In *18th Int. Software Product Lines Conference - Volume B , SPLC '14, Florence, Italy, Sept. 2014*, pages 44–51. ACM, 2014.

[IRBW16] N. Itzik, I. Reinhartz-Berger, and Y. Wand. Variability analysis of requirements: Considering behavioral differences and reflecting stakeholders perspectives. *IEEE Transactions on Software Engineering*, 42(7):687–706, 2016.

[KCH+90] Kyo C Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.

[LSS17] Yang Li, S. Schulze, and G. Saake. Reverse engineering variability from natural language documents: A systematic literature review. In *Proc. of the 21st International Systems and Software Product Line Conference-Volume A*, pages 133–142. ACM, 2017.

[MYC05] M. Moon, K. Yeom, and H. Seok Chae. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. *IEEE Transactions on Software Engineering*, 31(7):551–569, 2005.

[NBA+17] S. Ben Nasr, G. Bécan, M. Acher, J. B. Ferreira Filho, N. Sannier,B. Baudry, and J.-M. Davril. Automated extraction of product comparison matrices from informal product descriptions. *Journal of Systems and Software*, 124:82–103, 2017.

[PKS04] S. Park, M. Kim, and V. Sugumaran. A scenario, goal and feature-oriented domain analysis approach for developing software product lines. *Industrial Management & Data Systems*, 104(4):296–308, 2004.

[RFG+17] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, and S. Bacherini. Using nlp to detect requirements defects: An industrial experience in the railway domain. In *Int. Working Conf. on Requirements Engineering: Foundation for Software Quality*, pages 344–360. Springer, 2017.

[TB13] S. F. Tjong and D. M. Berry. The design of SREE - a prototype potential ambiguity finder for requirements specifications and lessons learned. In *Int. Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 80–95. Springer, 2013.

[tBFG18] M. H. ter Beek, A. Fantechi, and S. Gnesi. Product line models of large cyber-physical systems: the case of ERTMS/ETCS. In *Proc. of the 22nd Int. Conference on Systems and Software Product Line - Volume 1, SPLC 2018, Gothenburg, Sweden, September 10-14, 2018*, pages 208–214. ACM.

[tBFGS16] M. H. ter Beek, A. Fantechi, S. Gnesi, and L. Semini. Variability-based design of services for smart transportation systems. In *Proc. Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 7th International Symposium, ISoLA 2016, Corfu, Greece, Oct., 2016*, volume 9953 of *LNCS*, pages 465–481, 2016.

[vdML03] T. von der Maßen and H. Lichter. RequiLine: A requirements engineering tool for software product lines. In *Int. Workshop on Software Product-Family Engineering*, pages 168–180. Springer, 2003.

[YDRG+10] Hui Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh. Extending nocuous ambiguity analysis for anaphora in natural language requirements. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 25–34. IEEE, 2010.

[YWDRN10] Hui Yang, A. Willis, A. De Roeck, and B. Nuseibeh. Automatic detection of nocuous coordination ambiguities in natural language requirements. In *Proc. of the IEEE/ACM international conference on Automated software engineering*, pages 53–62. ACM, 2010.