Guiding Feature Models Synthesis from User-Stories: An Exploratory Approach

Thomas Georges
LIRMM, Univ Montpellier, CNRS
ITK - Predict & Decide
Montpellier, France
thomas.georges@lirmm.fr
thomas.georges@itk.fr

Liam Rice
LIRMM, Univ Montpellier, CNRS
Montpellier, France
liam.rice@lirmm.fr

Marianne Huchard
LIRMM, Univ Montpellier, CNRS
Montpellier, France
marianne.huchard@lirmm.fr

Mélanie König
ITK - Predict & Decide
Clapiers, France
melanie.konig@itk.fr

Clémentine Nebut
LIRMM, Univ Montpellier, CNRS
Montpellier, France
clementine.nebut@lirmm.fr

Chouki Tibermacine
LIRMM, Univ Montpellier, CNRS
Montpellier, France
chouki.tibermacine@lirmm.fr

ABSTRACT

User-stories are commonly used to define requirements in agile project management. In Software Product Lines (SPL), a user-story corresponds to a feature description (or part of it), that can be shared by several products. In practice, large SPL include a huge number of user-stories, making variability hard to grasp and handle. In this paper we present an exploratory approach that aims to guide the synthesis of Feature Models that capture and structure the commonalities and the variability expressed in these user-stories. The built Feature Models aim to help the project understanding, maintenance and evolution. Our approach first decomposes the user-stories to extract the roles and the features, using natural language processing techniques. In a second step, we group user-stories having the same topics thanks to a clustering method. This contributes to extract more general features. In a third step, we leverage the use of Formal Concept Analysis to extract logical constraints between the features that guide Feature Model synthesis. We illustrate our approach using a dataset from our industrial partner.

CCS CONCEPTS

• **Software and its engineering → Software reverse engineering**; **Software configuration management and version control systems**; **Agile software development**.

KEYWORDS

Software Product Line, Agile Process, User Story, Reengineering, SPL domain engineering, Feature Model, Natural Language Processing, Formal Concept Analysis

ACM Reference Format:
Thomas Georges, Liam Rice, Marianne Huchard, Mélanie König, Clémentine Nebut, and Chouki Tibermacine. 2023. Guiding Feature Models Synthesis from User-Stories: An Exploratory Approach. In *17th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS 2023), January 25–27, 2023, Odense, Denmark.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3571788.3571797

1 INTRODUCTION AND MOTIVATION

In modern IT projects, development teams organize themselves around Agile project specifications, like *epics* and *user-stories* distributed over sprints. Nowadays, many industrial projects are conducted based on this organization and leverage the use of platforms, like Github, Gitlab, Jira or Bitbucket. In parallel, Software Product Lines (SPL) are successfully introduced in a growing number of projects, for their qualities regarding the efficient development of similar products. Combining Agile approaches and SPL paradigm is challenging, raising organizational, technical and social questions [17]. Integrating Agile development methods in existing SPL has been considered for example in [18].

Our working context is a collaboration with an industrial partner (ITK[1]) which currently develops a family of similar applications for agriculture decision-making (e.g. yield forecast and disease prevention) with an Agile approach. The partner is willing to migrate to an SPL approach, thus considering an inverse problem compared to [18]. The partner builds its applications based on specifications written through user-stories and on a well documented code base, managed on a Gitlab server. As a step towards a migration to an SPL, our work aims at analyzing their large set of user-stories to evaluate if it is possible to use them for guiding a feature model synthesis [30, 31].

In this paper, we present our preliminary work conducted to achieve this goal. We have designed a process which takes as input a set of user-stories and which produces a set of logical constraints that aim to guide to potential feature models. This process goes through multiple steps in which user-stories are analyzed using Natural Language Processing (NLP) tools to identify

[1]https://www.itk.fr/en/

roles and **features** [35]. Then features are grouped into topics to identify more abstract features and to introduce a specialization hierarchy in the feature model. At last we leverage the use of Formal Concept Analysis [16] in order to identify and **structure variability** among the features depending on roles. We conjecture that these logical constraints are a relevant help in synthesizing feature models, as proposed in [9].

The remaining of the paper is organized as follows. In Section 2, we present the research questions tackled in this paper. We outline the proposed approach to answer these research questions in Section 3. In Section 4, we present a guide to build feature models from the produced logical constraints. We present the related work in Section 5, and then conclude in Section 6.

2 RESEARCH QUESTIONS

Before going into the details of the proposed solution, we present the research questions tackled in this exploratory work.

RQ. : **How can user-stories of a software family be used for guiding feature models synthesis?** We split this question into two sub-questions that respectively focus on identifying roles and (concrete) features, abstract features and logical constraints. In our industrial context, different roles do not share concrete features, but the abstract features may be shared. The logical constraints are to be used for guiding the building of a feature model.

RQ1. : **How to identify roles and features from user-stories?** Our partner actively uses the user-stories as the means for specifying the applications and maintaining this specification up-to-date. We then rely on these user-stories as a trusted source for finding the features of the partner applications. Most of these user-stories are composed of only two parts, respectively describing a role, and the functionality written from the point of view of this role. We hypothesize they correspond to 'features' in the meaning of SPLs.

RQ2. : **How to identify the logical constraints that could guide the feature model synthesis?** Once the features are identified for each application and role in this application, we dispose of a flat (tabular) description of the variability. This description implicitly contains the logical constraints, that we aim to identify and structure. Nevertheless, a feature model is not only a representation of a set of logical constraints. It is also built along ontological relationships (e.g. is-a, or refines) that our approach does not capture. In this work, we focus on assisting an expert in synthesizing a feature model by combining her ontological expertise and the identified logical constraints.

3 OUTLINE OF THE APPROACH

In order to answer the previous research questions, we have designed a process which is depicted in Figure 1. The process takes as input a set of user-stories of different products of the same "informal" family. These user-stories are analyzed throughout this process to produce at the end a set of structured logical constraints that aim to guide the building process of a feature model. In the following subsections, we detail the three steps of the process. Steps 1 and 2 answer RQ1; Step 3 answers RQ2.

Step 1. Processing User-Stories as Text Documents. User-stories are described using the following template, which is a quite common way of describing this kind of software requirements, as mentioned in [11]:
`"As a [persona], I [want to], [so that]."`

- `Persona` = the role that interacts with the application;
- `Want to` = the feature we want to identify;
- `So that` = the final goal of the user-story. In our partner user-stories, this part is not often described. We have thus simply ignored it in our work.

We use an illustrative example inspired by our partner applications[2].

It contains 12 user-stories, appearing in 3 decision-making software dedicated to Vine, Orchard and Almond crops respectively., e.g. US1 and US12 are:
`"As a Farmer, I can refresh the predicted weather."`
`"As an Admin, I can relaunch all failed simulations."`

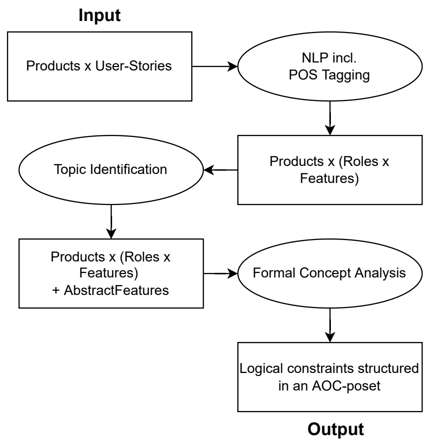
Input
Products x User-Stories
NLP incl. POS Tagging
Products x (Roles x Features)
Topic Identification
Products x (Roles x Features) + AbstractFeatures
Formal Concept Analysis
Logical constraints structured in an AOC-poset
Output

Figure 1: Proposed Process to Synthesize Feature Models from User-Stories.

In the first step, we analyze each user-story using NLP techniques, including tokenization, removal of stop words, lemmatization, and then POS (Part-Of-Speech) Tagging [6]. This analysis has two purposes: separate the role and the feature from the user-stories and prepare the clustering (Step 2). In Step 1, POS tagging allows us to separate roles from the initial set of features, e.g. US1 is separated into: `'Farmer'` (the role), and `'refresh the predicted weather'` (the feature). The role is extracted from a user-story by considering the "*first nominal group*" as a POS tag. We then distribute the features in different sets, each set corresponding to a role. Table 1 schematizes the distribution of the 12 user-stories of our example over the products and the roles. The remaining steps separately apply on each set of features (one per role).

Step 2. Topic Modeling. In order to have a richer set of features, we leverage the use of a clustering method, currently K-Means [7], to identify "**Topics**". These topics aim to represent more abstract

[2]The example can be shown at: https://gite.lirmm.fr/tgeorges/artefacts-guiding-feature-models-synthesis-from-user-stories-an-exploratory-approach

Table 1: Schematizing an example: 12 user-stories from 3 products, distributed along the identified roles. Features, products and roles after NLP. Row *i* corresponds to the user-story *i* without the role and the part *"I can", "I want to be able", "I know"*.

		Products			Roles	
Features	Feat. no.	Vine	Orchard	Almond	Farmer	Admin
refresh the predicted weather	1	x	x		x	
CRUD plots	2	x	x	x	x	
edit the parameters of a plot (current season)	3	x	x		x	
sort my plots in the list	4		x		x	
filter my plots in the list	5		x		x	
export observation data for a plot	6	x			x	
know when my plots will be in danger	7	x	x		x	
manage my irrigations and recommendations in my favorite unit	8			x	x	
choose my preferred irrigation unit in my user-settings	9	x			x	
view my irrigation recommendations in my favorite unit	10	x		x	x	
CRUD a farmer	11	x	x			x
relaunch all failed simulations	12	x	x	x		x

features that may appear later as intermediate nodes in the synthesized feature models. We start from the set of features identified in the user-stories. At this point, the user-stories are tokenized, lemmatized and POS tagged to get a better result during the clustering. The following user-story *"As a Farmer, I can manage my irrigations and recommendations in my favorite unit"* is transformed into: *manage irrigations recommendations favorite unit.* The Part of speech tagging provides as an output: "`manage_VB irrigations_NNS recommendations_NNS favorite_JJ unit_NN`", with "`VB`" stands for a verb, "`NN`" for a noun, "`NNS`" for a plural noun and "`JJ`" for an adjective.

Since clustering methods require the choice of a number of clusters that we cannot know *a priori*, we used Elbow method [32] to identify the optimal number of clusters for our data. This method measures the distribution of our input data and tries to optimize the number of clusters so that intra-cluster cohesion is maximized.

Elbow and K-Means need data represented as numbers. In our case, we provide these methods with vector representations of features. For doing so, we have used Doc2Vec [20] to vectorize the extracted features. The vectorization puts the user-stories to a d-dimension vector (d is the number of user-stories), like for example `<0.4125, -1.6098, 0.6047, ... ,-1.4257, -1.2321>`. Then, K-Means uses euclidean distance to group the nearest user-stories to form clusters.

Given a set of observations (x1, x2, ..., xn), where each observation is a d-dimensional real vector, k-Means clustering aims to partition the n observations into k ($\leq$ n) sets S = {S1,S2, ...,Sk} so as to minimize the within-cluster sum of squares.

The choice of Elbow, K-Means and Doc2Vec is motivated by the good results obtained with them in existing works [4, 12], their complete documentation and the existence of an effective tool support. K-Means is a common used clustering method. In our case it was the most effective method, compared to other methods that we have experimented on our dataset, including LSA/LSI and LDA.

The topics obtained for our example are depicted in Table 2. They appear in the 'AbstractFeatures' column. Currently the abstract features have no automatically generated name. This is left as future work and has been done manually for our example.

Step 3. Logical constraints identification and structuring in an AOC-poset. In the third step, we use Formal Concept Analysis (FCA) for highlighting and structuring the logical constraints, similarly to [9]. Information obtained after topic modeling (e.g. Table 2) is split into as many tables as there are roles. These new tables (called *formal contexts* according to the vocabulary of FCA) describe the different products by their identified features and abstract features. The formal context is shown for the role `Farmer`

in Table 3. *Formal concepts* can be extracted from these formal contexts. A formal concept is a maximal group of products (extent) associated with the maximal group of features (intent) they share. For example $\{Orchard, Vine, Almond\}, \{(2), PlotManagement\}$ is a concept that can be extracted from Table 3. The formal concepts of a formal context can be structured with a specialization order based on their intent inclusion. This leads to a hierarchical representation, as shown in Figure 2. The representation is simplified so that a feature (resp. a product) of a concept is inherited in its sub-concepts (resp. super-concepts). We restrict here the concepts to those that introduce a product or a feature. The resulting hierarchy is called an AOC-poset[3]. The AOC-Poset can be interpreted into a propositional logic formula which is equivalent to the one expressed in the Formal Context [8]. In this AOC-poset, we can visually read mandatory features, feature co-occurrences, mutex and implications, and candidate OR and XOR groups as explained in [9]. For example, Feature `CRUD ...(2)` , being in Top Concept 0, is a mandatory feature. Feature `manage my irrigations and recommendations in my favorite unit (8)` (in sub-concept 4) *implies* Feature `view my irrigation recommendations in my favorite unit (10)` (in super-concept 5). Other constraints are discussed in Section 4.

Implementation. We are developing a prototype tool to implement the previous process. Our preprocessing and topic modeling are implemented in Python thanks to the large number of available well documented tools and libraries. For example *Gensim* has been used for topic modeling and *nltk* for NLP. Formal Concept Analysis tasks are implemented using *Cogui* [4], a tool for graph based knowledge representation and management that also implements FCA. Finally editing the feature model is done in the *FeatureIDE* plugin for *Eclipse*[5].

4 GUIDING THE BUILDING OF A FEATURE MODEL FROM THE AOC-POSET

In this section, we aim to highlight the potential of the approach for guiding experts during a feature model synthesis. The guidance begins with the supply of the AOC-Poset depicted in Figure 2 and ends with a feature model as shown in Figure 3. We hereafter present examples of the guidance process.

We observe in the AOC-Poset that the abstract feature *Plot Management* and the feature *CRUD plots (2)* are shared by all the products (mandatory), as they are both introduced in the top concept.

[3]AOC-poset stands for a *Partially Ordered Set of Attribute Object introducing Concepts (concepts that introduce an attribute or an object).*
[4]https://www.lirmm.fr/cogui/
[5]https://featureide.github.io/

Table 2: Data after topic modeling: The columns AbstractFeatures have been added to Table 1. `Weather`, `Plot` and `Irrigation` are short names in the text for `WeatherManagement`, `PlotManagement` and `IrrigationManagement`.

		Products			Roles		AbstractFeatures				
Features	Features no.	Vine	Orchard	Almond	Farmer	Admin	Weather Management	Plot Management	Irrigation Management	Simulation Management	Farmer Management
refresh the predicted weather	1	x	x		x		x				
CRUD plots	2	x	x	x	x			x			
edit the parameters of a plot (current season)	3	x	x		x			x			
sort my plots in the list	4		x		x			x			
filter my plots in the list	5		x		x			x			
export observation data for a plot	6	x			x			x			
know when my plots will be in danger	7	x	x		x			x			
manage my irrigations and recommendations in my favorite unit	8			x	x				x		
choose my preferred irrigation unit in my user-settings	9	x			x				x		
view my irrigation recommendations in my favorite unit	10	x		x	x				x		
CRUD a farmer	11	x	x			x					x
relaunch all failed simulations	12	x	x	x		x				x	

Table 3: Formal Context for the *Farmer* role.

Farmer	Features										AbstractFeatures		
	1	2	3	4	5	6	7	8	9	10	Weather	Plot	Irrigation
Vine	x	x	x			x	x		x	x	x	x	x
Orchard	x	x	x	x	x		x				x	x	
Almond		x						x		x		x	x

One of them could be used to root the feature model or the experts could consider that none of them is the root. Let us suppose that they choose the second solution, rooting the feature model in a general feature called *FeaturesOfFarmer*. Below the root, the two mandatory features should appear. They can be positioned at the same level, or not. The expert may know that *CRUD plots* is a specialization of the abstract feature *PlotManagement*, or this could be deduced from the fact that *CRUD plots* is an element of the group *PlotManagement* formed at Step 2.

The abstract features *WeatherManagement* and *IrrigationManagement* are introduced in the sibling Concepts 1 and 5. The extents of Concepts 1 and 5 intersect (Vine is a common product of both), and their union covers the extent of Concept 0. This suggests that *WeatherManagement* and *IrrigationManagement* are (1) either two optional features, (2) or children in an OR group. These optional or children features can be rooted (a) in `FeaturesOfFarmer`, (b) in `PlotManagement`, (c) in `CRUD a plot`. Let us consider that the experts choose solution (1)(a). Feature 2 (`CRUD a plot`) and other features related to plots (3,4,5,6,7) are attached by the experts below `PlotManagement`. Features 1, 3, 4, 5, 6, 7, 8, 9, 10 are optional features as they are not shared by the 3 products. Other information can be extracted from the AOC-poset for these features due to their position in the concepts. For example, *sort my plots in the list (4)* and *filter my plots in the list (5)* appear in the same concept, belong to the same abstract feature and are co-occurring. This can be used to generate the crosscutting constraint: *sort my plots in the list* ⇔ *filter my plots in the list* which is shown below the feature model of Figure 3. For two features from the same concept extent without the same abstract feature, the crosscutting constraints are more susceptible to be accidental. In all cases, the experts have to assess these constraints.

Concept 5 and its sub-concepts 3 and 4 introduce, besides the abstract feature *IrrigationManagement*, concrete features that refine it (features 6, 8, 9 ,10), and feature 6 which is related to plots. As features 8, 9 and 10 were grouped at Step 2 in the group called *IrrigationManagement*, and at least one of them should be present in a product, the experts may choose to make them children in an OR group. The AOC-poset indicates that every feature among 6, 8 and 9 implies feature 10. The experts are free to use this information or to consider that these are accidental implications observed due to the low number of analyzed products (only 3).

Concept 1 introduces the abstract feature *WeatherManagement* and the three concrete features it groups (features 1, 3, 7). As the examined products contain at least one of these 3 concrete features, the experts are inclined to make them children in an OR group in the feature model rooted at *WeatherManagement*.

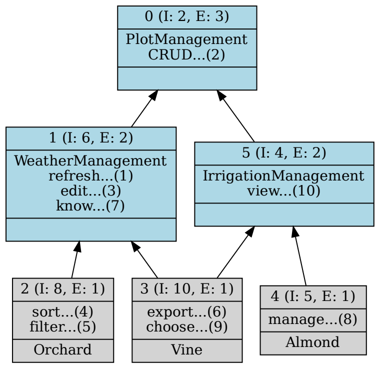

Figure 2: The Farmer AOC-Poset.

We launched the complete process on 127 formatted user-stories from our industrial partner. They describe the features of 8 products.

The number of clusters determined by the Elbow method ranges between 10 and 15 with 10 features on each on average. At first glance, the formed clusters share similar features, but we need to analyze thoroughly the obtained results. We plan in the near future to evaluate the quality of the obtained clusters with our industrial partner, and to build the feature model with them, as described previously.

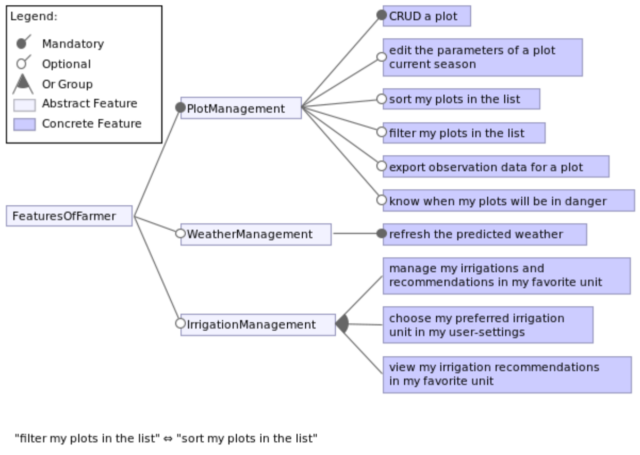
Legend:
Mandatory
Optional
Or Group
Abstract Feature
Concrete Feature
FeaturesOfFarmer
PlotManagement
WeatherManagement
IrrigationManagement
CRUD a plot
edit the parameters of a plot current season
sort my plots in the list
filter my plots in the list
export observation data for a plot
know when my plots will be in danger
refresh the predicted weather
manage my irrigations and recommendations in my favorite unit
choose my preferred irrigation unit in my user-settings
view my irrigation recommendations in my favorite unit
"filter my plots in the list" ⇔ "sort my plots in the list"

Figure 3: One possible Farmer Feature Model.

We are aware that the construction of the feature model requires some work from the domain expert, who is guided by the AOC-Poset. First she/he needs to identify the logical constraints to taking them into consideration to build a consistent feature model. The interpretation of the AOC-Poset is required from the domain expert, and this may require some effort from a beginner with FCA. In a future work, we plan to evaluate the overhead of this interpretation by measuring the efforts necessary to produce the feature model from the AOC-Poset. We also envisage to provide an assistance in this interpretation, based for example on the work of Jessie Galasso-Carbonnel et al. on equivalence classes of feature models in [9].

5 RELATED WORK

In our work we have been inspired by several similar approaches, like the use of NLP to exploit user-stories to generate UML use case diagrams [23] or to generate UML sequence diagrams [14].

The work in [6] leverages the use of topic modeling and POS-tagging, as in our work, for supporting creativity in Requirement engineering. In contrast to our work, this approach does not deal with user-stories and does not consider feature models.

Some related works [24, 33] assessed the effectiveness of user-stories in software development processes. They however do not address the specific context of software product lines. A focus on Requirement engineering in a software product lines context has been developed in [3]. For pre-processing, the authors use NLP techniques equivalent to those used in our process. But, in contrast to our work, they do not consider user-stories, but another type of requirements. The interest of an automatic technique for requirement analysis with natural language processing is shown in [2]. We used some equivalent methods and tools to transform textual specifications into visual models as detailed in [26] to design our approach. A Systematic Literature Review (SLR) on 13 studies highlights the systematic use of NPL techniques for feature identification on requirements described in natural language [5]. We tried some of the proposed techniques (e.g. POS tagging, LSA, LDA, k-means) and it would be relevant to consider some others in future work (e.g. hierarchical agglomerative tagging).

Formal concept analysis has been used in software product lines for addressing several tasks, e.g. for requirements [25], for logical constraints extraction [15, 22], feature model synthesis [27], for feature-to-code traceability [28], recovering SPL architecture [29], or for feature location in [1, 34].

To complete our process, in a future work, we plan to apply feature location techniques based on agile requirements. Some techniques that may inspire us are detailed in [10, 13, 21]. One of our future works is also to improve our topic modeling by the use of an ontology as in [19].

6 CONCLUSION

We presented in this paper a preliminary research work on the analysis of Agile software specifications (in our case user-stories) of a product family, to identify features and guide feature model synthesis. For doing so, we designed a process that leverages the use of a variety of techniques, including NLP ones, Clustering and Formal Concept Analysis. We illustrated this process on a small set of user-stories inspired by a real-world project of our industrial partner. In the near future, we plan to evaluate the approach by taking into consideration the whole dataset of user-stories of our industrial partner, in addition to other datasets of user-stories from open-source repositories. We will probably face some scalability and "dirty data" issues with such large datasets. We need in these cases to proceed to some decomposition and cleansing tasks upfront in the process. We will probably also need an ontology to represent the vocabulary of the domain of user-stories to better identify clusters.

We also will design a methodology and a tool, inspired by [9], to assist the experts in feature model synthesis. As a perspective to this work, we plan to analyze other useful information that accompany user-stories, like commit contents, merge requests and technical issues in order to refine feature model synthesis and then to proceed to feature location.

ACKNOWLEDGMENT

This work is funded by ITK and has been partially supported by the ANR SmartFCA project Grant ANR-21-CE23-0023 of the French National Research Agency.

REFERENCES

[1] Ra'Fat Al-Msie'deen, Abdelhak Seriai, Marianne Huchard, Christelle Urtado, Sylvain Vauttier, and Hamzeh Eyal Salman. 2013. Feature Location in a Collection of Software Product Variants Using Formal Concept Analysis. In *Safe and Secure Software Reuse - 13th International Conference on Software Reuse, ICSR 2013, Pisa, Italy, June 18-20. Proceedings (Lecture Notes in Computer Science, Vol. 7925)*, John M. Favaro and Maurizio Morisio (Eds.). Springer, 302–307. https://doi.org/10.1007/978-3-642-38977-1_22

[2] Vincenzo Ambriola and Vincenzo Gervasi. 1997. Processing Natural Language Requirements. In *1997 International Conference on Automated Software Engineering, ASE 1997, Lake Tahoe, CA, USA, November 2-5, 1997*. IEEE Computer Society, 36–45. https://doi.org/10.1109/ASE.1997.632822

[3] Maximiliano Arias, Agustina Buccella, and Alejandra Cechich. 2018. A Framework for Managing Requirements of Software Product Lines. *Electronic Notes in*

Theoretical Computer Science 339 (2018), 5–20. https://doi.org/10.1016/j.entcs.2018.06.002 The XLII Latin American Computing Conference.

[4] Wesley Klewerton Guez Assunção and Silvia Regina Vergilio. 2014. Feature location for software product line migration: a mapping study. In *18th International Software Product Lines Conference - Companion Volume for Workshop, Tools and Demo papers, SPLC '14, Florence, Italy, September 15-19, 2014*, Stefania Gnesi, Alessandro Fantechi, Maurice H. ter Beek, Goetz Botterweck, and Martin Becker (Eds.). ACM, 52–59. https://doi.org/10.1145/2647908.2655967

[5] Noor Hasrina Bakar, Zarinah M. Kasirun, and Norsaremah Salleh. 2015. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software* 106 (2015), 132–149. https://doi.org/10.1016/j.jss.2015.05.006

[6] Tanmay Bhowmik, Nan Niu, Juha Savolainen, and Anas Mahmoud. 2015. Leveraging topic modeling and part-of-speech tagging to support combinational creativity in requirements engineering. *Requirements Engineering* 20 (04 2015). https://doi.org/10.1007/s00766-015-0226-2

[7] Hans-Hermann Bock. 2007. *Clustering Methods: A History of k-Means Algorithms.* Springer Berlin Heidelberg, Berlin, Heidelberg, 161–172. https://doi.org/10.1007/978-3-540-73560-1_15

[8] Jessie Carbonnel, David Delahaye, Marianne Huchard, and Clémentine Nebut. 2019. Graph-Based Variability Modelling: Towards a Classification of Existing Formalisms. In *Graph-Based Representation and Reasoning - 24th International Conference on Conceptual Structures, ICCS 2019, Marburg, Germany, July 1-4, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11530)*, Dominik Endres, Mehwish Alam, and Diana Sotropa (Eds.). Springer, 27–41. https://doi.org/10.1007/978-3-030-23182-8_3

[9] Jessie Carbonnel, Marianne Huchard, and Clémentine Nebut. 2019. Modelling equivalence classes of feature models with concept lattices to assist their extraction from product descriptions. *Journal of Systems and Software* 152 (2019), 1–23. https://doi.org/10.1016/j.jss.2019.02.027

[10] Kunrong Chen and Václav Rajlich. 2000. Case Study of Feature Location Using Dependence Graph. In *8th International Workshop on Program Comprehension (IWPC 2000), 10-11 June 2000, Limerick, Ireland.* IEEE Computer Society, 241–247. https://doi.org/10.1109/WPC.2000.852498

[11] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development.* Addison Wesley Longman Publishing Co., Inc., USA.

[12] Daniel Cruz, Eduardo Figueiredo, and Jabier Martinez. 2019. A Literature Review and Comparison of Three Feature Location Techniques Using ArgoUML-SPL. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems* (Leuven, Belgium) *(VAMOS '19).* Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. https://doi.org/10.1145/3302333.3302343

[13] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature location in source code: A taxonomy and survey. *Journal of Software Maintenance and Evolution: Research and Practice* 25 (01 2013). https://doi.org/10.1002/smr.567

[14] Meryem Elallaoui, Khalid Nafil, and Raja Touahni. 2015. Automatic generation of UML sequence diagrams from user stories in Scrum process. In *10th International Conference on Intelligent Systems: Theories and Applications, SITA 2015, Rabat, Morocco, October 20-21, 2015.* IEEE, 1–6. https://doi.org/10.1109/SITA.2015.7358415

[15] Jessie Galasso and Marianne Huchard. 2023. Extending Boolean Variability Relationship Extraction to Multi-valued Software Descriptions. In *Handbook of Re-Engineering Software Intensive Systems into Software Product Lines*, Roberto E. Lopez-Herrejon, Jabier Martinez, Wesley Klewerton Guez Assunção, Tewfik Ziadi, Mathieu Acher, and Silvia Vergilio (Eds.). Springer International Publishing, 143–173. https://doi.org/10.1007/978-3-031-11686-5_6

[16] Bernhard Ganter and Rudolf Wille. 1999. *Formal Concept Analysis - Mathematical Foundations.* Springer. https://doi.org/10.1007/978-3-642-59830-2

[17] Philipp Hohl, Jürgen Münch, Kurt Schneider, and Michael Stupperich. 2017. Real-Life Challenges on Agile Software Product Lines in Automotive. In *Product-Focused Software Process Improvement - 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29 - December 1, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10611)*, Michael Felderer, Daniel Méndez Fernández, Burak Turhan, Marcos Kalinowski, Federica Sarro, and Dietmar Winkler (Eds.). Springer, 28–36. https://doi.org/10.1007/978-3-319-69926-4_3

[18] Jil Ann-Christin Klünder, Philipp Hohl, Nils Prenner, and Kurt Schneider. 2019. Transformation towards agile software product line engineering in large companies: A literature review. *J. Softw. Evol. Process.* 31, 5 (2019). https://doi.org/10.1002/smr.2168

[19] Sven Koerner and Torbenbrumm. 2011. NATURAL LANGUAGE SPECIFICATION IMPROVEMENT WITH ONTOLOGIES. *International Journal of Semantic Computing* 03 (11 2011). https://doi.org/10.1142/S1793351X09000872

[20] Quoc V. Le and Tomás Mikolov. 2014. Distributed Representations of Sentences and Documents. *CoRR* abs/1405.4053 (2014). arXiv:1405.4053 http://arxiv.org/abs/1405.4053

[21] Adrian Lienhard, Orla Greevy, and Oscar Nierstrasz. 2007. Tracking Objects to Detect Feature Dependencies. In *15th International Conference on Program Comprehension (ICPC 2007), June 26-29, 2007, Banff, Alberta, Canada.* IEEE Computer Society, 59–68. https://doi.org/10.1109/ICPC.2007.38

[22] Felix Loesch and Erhard Ploedereder. 2007. Restructuring Variability in Software Product Lines using Concept Analysis of Product Configurations. In *11th European Conference on Software Maintenance and Reengineering, Software Evolution in Complex Software Intensive Systems, CSMR 2007, 21-23 March 2007, Amsterdam, The Netherlands*, René L. Krikhaar, Chris Verhoef, and Giuseppe A. Di Lucca (Eds.). IEEE Computer Society, 159–170. https://doi.org/10.1109/CSMR.2007.40

[23] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn Van der Werf, and Sjaak Brinkkemper. 2016. Improving agile requirements: the Quality User Story framework and tool. *Requirements Engineering* 21 (09 2016). https://doi.org/10.1007/s00766-016-0250-x

[24] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. 2016. The Use and Effectiveness of User Stories in Practice. In *Requirements Engineering: Foundation for Software Quality - 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings (Lecture Notes in Computer Science, Vol. 9619)*, Maya Daneva and Oscar Pastor (Eds.). Springer, 205–222. https://doi.org/10.1007/978-3-319-30282-9_14

[25] Nan Niu and Steve M. Easterbrook. 2009. Concept analysis for product line requirements. In *Proceedings of the 8th International Conference on Aspect-Oriented Software Development, AOSD 2009, Charlottesville, Virginia, USA, March 2-6, 2009*, Kevin J. Sullivan, Ana Moreira, Christa Schwanninger, and Jeff Gray (Eds.). ACM, 137–148. https://doi.org/10.1145/1509239.1509259

[26] Cristina-Claudia Osman and Paula Zalhan. 2016. From Natural Language Text to Visual Models: A survey of Issues and Approaches. *Informatica Economica* 20 (12 2016), 44–61. https://doi.org/10.12948/issn14531305/20.4.2016.05

[27] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. 2011. Extraction of feature models from formal contexts. In *Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22-26, 2011. Workshop Proceedings (Volume 2)*, Ina Schaefer, Isabel John, and Klaus Schmid (Eds.). ACM, 4. https://doi.org/10.1145/2019136.2019141

[28] Hamzeh Eyal Salman, Abdelhak-Djamel Seriai, and Christophe Dony. 2013. Feature-to-code traceability in a collection of software variants: Combining formal concept analysis and information retrieval. In *IEEE 14th International Conference on Information Reuse & Integration, IRI 2013, San Francisco, CA, USA, August 14-16, 2013.* IEEE Computer Society, 209–216. https://doi.org/10.1109/IRI.2013.6642474

[29] Anas Shatnawi, Abdelhak-Djamel Seriai, and Houari A. Sahraoui. 2017. Recovering software product line architecture of a family of object-oriented product variants. *J. Syst. Softw.* 131 (2017), 325–346. https://doi.org/10.1016/j.jss.2016.07.039

[30] Steven She, Krzysztof Czarnecki, and Andrzej Wąsowski. 2012. Usage Scenarios for Feature Model Synthesis. In *Proceedings of the VARiability for You Workshop: Variability Modeling Made Useful for Everyone* (Innsbruck, Austria) *(VARY '12).* Association for Computing Machinery, New York, NY, USA, 15–20. https://doi.org/10.1145/2425415.2425419

[31] She, Steven. 2013. *Feature Model Synthesis.* Ph. D. Dissertation. University of Waterloo. http://hdl.handle.net/10012/7834

[32] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the Number of Clusters in a Data Set Via the Gap Statistic. *Journal of the Royal Statistical Society Series B* 63 (02 2001), 411–423. https://doi.org/10.1111/1467-9868.00293

[33] Yves Wautelet, Samedi Heng, Manuel Kolp, and Isabelle Mirbel. 2014. Unifying and Extending User Story Models. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8484)*, Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos, Haralambos Mouratidis, and Jennifer Horkoff (Eds.). Springer, 211–225. https://doi.org/10.1007/978-3-319-07881-6_15

[34] Yinxing Xue, Zhenchang Xing, and Stan Jarzabek. 2012. Feature Location in a Collection of Product Variants. In *19th Working Conference on Reverse Engineering, WCRE 2012, Kingston, ON, Canada, October 15-18, 2012.* IEEE Computer Society, 145–154. https://doi.org/10.1109/WCRE.2012.24

[35] Pamela Zave. 2003. *An experiment in feature engineering.* Springer New York, New York, NY, 353–377. https://doi.org/10.1007/978-0-387-21798-7_17