Feature and Variability Extraction from Agile Specifications and their Related Source Code for Software Product Line Migration

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

ABSTRACT

Migrating a set of similar software products into a Software Product Line is a time-consuming and costly process which, ultimately, provides an important gain in time and customization. Conducting this migration within an agile development process is a complex process which requires discipline and adaptation. We think it can be beneficial to drive the migration by leveraging agile software specifications and the source code versioning platform. Currently, we are working on a method, whose design is explained in this paper, which exploits: (1) Epics and User stories to identify features and variability and (2) the source code associated to code merges related to User stories and Epics to locate them. We plan to extract features and variability inside Epics and User stories using Natural Language Processing (NLP) techniques. Then we plan to investigate how formal concept analysis (FCA) and relational concept analysis (RCA) can assist feature model synthesis and establish mappings between features and source code. These knowledge discovery methods have been chosen for their ability to highlight and hierarchically organize groups of similar artefacts. FCA only considers artefact description to establish groups of similar artefacts. RCA groups similarly described artefacts that, in addition, share similar relationships to other artefact groups. We also plan to evaluate the method within the context of a company (ITK) with which we collaborate, using its code base and the associated project management artifacts. We also will assess how the method can be generalized to public projects in source code versioning platforms.

ACM Reference Format:
Thomas Georges. 2022. Feature and Variability Extraction from Agile Specifications and their Related Source Code for Software Product Line Migration. In *26th ACM International Systems and Software Product Line Conference - Volume B (SPLC '22), September 12–16, 2022, Graz, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3503229.3547065

1 INTRODUCTION AND MOTIVATION

A Software Product Line is a game changing tool for companies who invest enough on it. When a set of similar software is already developed, migrating this set into a Software Product Line requires investment in time and resource [17]. Our industrial partner wants to achieve this kind of migration from a set of software that is developed on a common code base, as it faces an important need to improve software customization and time to market.

The goal of the thesis is to support the migration process by developing analysis and refactoring methods and tools. Main activities in this migration are feature identification and location. The company's development process is based on *Agility*. It follows many software engineering good practices like versioning, branching, sprints and User stories for requirement specification.

The challenge here is to support the complete process of migration while keeping the company's agile process. Feature identification from agile specification is motivated by the need to integrate the current development process into the migration, and potentially continue building the software product line incrementally.

We are currently working on the identification of relations between source code and agile specifications through code merges and User stories. In the company projects, User stories are well-written and rigorously specified; they thereby reflect the code base features. Code merges are therefore the link between the source code and the features in the user stories. In this exploratory study, we plan to analyse User stories to identify features and their associated source code, via code merges, for performing feature location and extracting feature variability.

The remainder of the paper is organized as follows. Section 2 presents the research questions. Section 3 exposes our research methodology. Section 4 introduces the preliminary results. Section 5 details the work plan and Section 6 concludes the paper.

2 RESEARCH QUESTIONS

User stories and by extension, Epics, are specifications commonly used in an agile processes [11]. We will call them *agile specifications* in the remainder of this paper. User stories and Epics implicitly document the features the client expresses or has in mind. The term *feature* refers to functional and non-functional characteristics of a system [1]. The goal of this exploratory study is to investigate ways to exploit the agile specifications to, firstly, identify features and feature variability (*feature model synthesis*), and secondly, locate features in the source code, in order to assist the migration process. We investigate the use of Formal Concept Analysis and Relational Concept Analysis, two methods which already proved their efficiency for assisting feature model synthesis [3] and feature location [14].

In our work we address the following research questions:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SPLC '22, September 12–16, 2022, Graz, Austria
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9206-8/22/09…$15.00
https://doi.org/10.1145/3503229.3547065

RQ1. : Can we assist the migration steps of (1) feature model synthesis and (2) traceability between features and existing code, by using agile specifications extracted from the issue tracking system of the version control platform (Gitlab, in our case), in addition to the source code?

First, feature model synthesis is conducted by separating "Roles" and "Features" from User stories. Then, we will investigate the traceability between features and existing code, through merges, to complete the feature model.

This research question is decomposed into two sub-questions.

RQ1.1. : Can we leverage NLP to identify features in the User stories and FCA to assist feature model synthesis?

The idea is to look at User stories to find features, and separate them by roles [16]. This would be done by analyzing text with Natural Language Processing (NLP) Techniques to separate the role part from the action part. Then, by considering the products and the features extracted from their User stories, and based on an existing previous work from our team [5], we will design a complete methodology to use FCA to derive a feature model.

RQ1.2. : Can we leverage Relational Concept Analysis (RCA) to obtain traceability between features and existing source code using the issue tracking system and code merge?

Our idea is to group features and to group code-components so that a code-component group corresponds to the implementation of a feature group and vice versa. This corresponds to a match between feature variability and code variability. This will complete the feature model by feature location information, i.e. information on the source code implementing a feature.

3 RESEARCH METHODOLOGY AND APPROACH

To address the research questions, we started with a cartography of collaborative version control applications, followed by a study of the interest of Relational Concept Analysis and Formal Concept Analysis on our case, as there were already successful applications of these approaches in Software Product Line reverse engineering.

As an input of our process, we consider a code base hosted in a source code versioning platform, like *GitLab* or *GitHub*. We also consider a set of User stories defined in this versioning platform, as a set of issues, for which developers have created merge/pull-requests. The User stories are connected to the source code with *Code Merge*.

In our research plan, we first identify the Source Code associated to a feature expressed in a User story then we identify variability on the User stories. The final goal is a feature model representing the extracted variability.

Our industrial collaborators develop and manage their projects on the collaborative source code versioning platform *Gitlab* where we will experiment the feasibility of our approach. Nevertheless, we will design our approach so that it could be applied to any collaborative version control platform.

Collaborative version control platform cartography: Collaborative version control platforms are powerful complete tools for software engineers. These platforms are familiar to any agile development team. They are used for their issue and bug tracking, continuous integration, documentation and wikis for development projects.

We are working a part of these large tools, mainly on the issue tracking part. We focus more specifically on Stories which are Epics or User stories and their relationship, by technical issues and associated Code Merges, to the source code.

As shown in Figure 1:

- An Epic is composed of a set of User stories. This could be seen as a super User story.
- A User story is a feature defined by a product owner and expressed initially by the end User. It is formatted as follows: *"As a Producer, I can visualize my plots.* The User story is composed of three parts : Role (e.g. *"Producer"*), Action (e.g. *"Can visualize"*) and complements (e.g. *"my plots*).

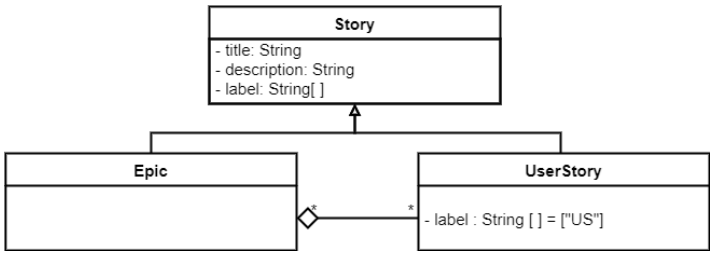

Figure 1: A Model of User stories

As detailed in Figure 2:

- A **Story** may be composed of technical issues with code details.
- A **Merge** may compose the stories or technical issues. A Merge is an aggregation of code changes which are an *addition*, a *deletion* or a *suppression* in a source code file. The term Merge comes from two branches in a git flow that have to be mixed or merged.

Elaborating the (meta-)model representing these concepts was the first step to explore our ideas.

Relational Concept Analysis (RCA) *and* ***Formal Concept Analysis FCA***. They are both knowledge representation and extraction methods. These two methods are especially well adapted for variability extraction and representation.

- Formal Concept Analysis builds a concept lattice. Concepts group objects sharing attributes. They are extracted from a table, called a Formal Context [7] where objects are described by their attributes. In our case we use the AOC-Poset, a sub-order of the concept lattice, which contains all feature variability information. The AOC-Poset has the capability of being used to derive Feature Models [4].
- Relational Concept Analysis combines Formal Contexts with Relational Contexts [9]. It is used to classify objects of several categories upon shared attributes and shared relationships to other objects (from the same or from another category).

The features traceability as studied in [10] is in our case supported by FCA and RCA.

In Figure 3 and Figure 4, each class induces a Formal Context and each association a Relational Context. A User story is described

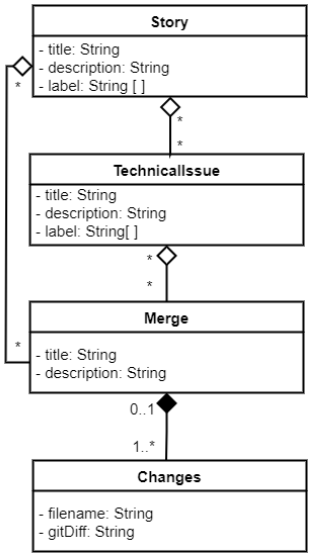

Figure 2: Relationships between User stories and other Concepts

with its title composed of a Role, an Action and its complement. The User story is also composed of Merges.

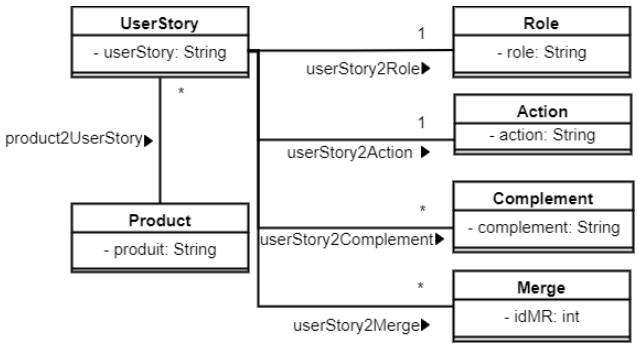

Figure 3: Collaborative version control platforms Meta model – Part 1

A Merge is composed of changes on source code files. A change could be a creation, a modification or a deletion. We stop our granularity at the file level but we plan in a future work to go farther and analyse source code elements (classes, functions/methods, statements, etc.).

Natural language processing (NLP). We will apply NLP techniques on the User stories to separate the *role* and the feature for assisting feature model synthesis and to enrich the variability description. Variability groups formed by clustering methods will be added to the existing groups formed with Relational Concept Analysis.

The idea with *NLP* is to divide the User story into the actor role and the feature. The User stories have to be correctly formatted to ensure the success of this step. We assume that the User stories are

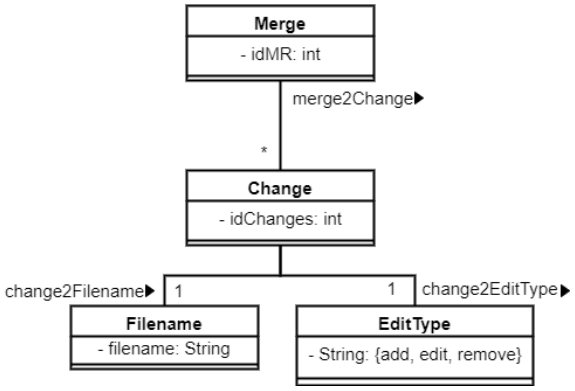

Figure 4: Collaborative version control platforms Meta model – Part 2

structured as follows: *"As a [persona], I [want to], [so that]."*. Already some work was done in the literature to exploit the User stories, for example to transform User stories into *Use-cases* [12].

For variability identification from similar issues, some methods will be studied to classify and extract as much variability as possible, for example to study the similarity of User stories:

- Latent Dirichlet allocation [2]
- Latent semantic analysis [6] and Latent Semantic Indexing [15]
- word2vec [13]

4 PRELIMINARY RESULTS

Our approach already gives encouraging results on a small data set. In the following example with only four User stories, we can find groups and variability. Larger experiments are planned in a near future.

Using FCA and RCA provides exhaustive variability extraction and visualization in the concept lattices. Creating groups of objects sharing attributes and relations is a specificity that concluded in us using FCA and RCA.

A small example to explore variability. Let us consider these four User stories :

(1) As a farmer, I can create a plot in my farm
(2) As a farmer, I can visualize my plot of my farm
(3) As a counselor, I can manage data on my farm list
(4) As a counselor, I can visualize for each plot from a farm in my farm list

These four User stories share similar complements but with a different action *(Can create*, *Can visualize* and *Can manage*). Two User stories share the role *farmer* and two others the role *counselor*. We are already able to detect and visualize variability on the User stories on this example. Nevertheless it is possible to go further using the source code.

Consider the following code Merges *M1*, *M2*, *M3* and *M4* with two changes for each one: *ch1.1 and ch1.2*, *ch2.1 and ch2.2* and *ch3.1 and ch3.2* and *ch4.1 and ch4.2* respectively.

We are interested in the changes from our Merges :

- *ch1.1 create* the file *plot*

- *ch1.2 create* the file *farm*
- *ch2.1 create* the file *view*
- *ch2.2 edit* the file *plot*
- *ch3.1 edit* the file *farm*
- *ch3.2 create* the file *counselor*
- *ch4.1 edit* the file *counselor*
- *ch4.2 edit* the file *view*

We have four features expressed from the four User stories. With only this small example we already have trivial relations between files and User stories from the study of the merges. We also already see the complexity from just four User stories, that is why we use FCA and RCA, to systematically build groups of objects sharing concepts.

Concept lattices are organized with a specialization relation. A concept (group) gathers objects (inherited by its super-concepts) and attributes (inherited by its sub-concepts). Thanks to the highlighting of the shared relations, RCA also allows navigation between concepts (groups) of objects from different categories (classes). For example, in our data the group composed of the objects *Farmer*, and *Plot* share the object *Farm* with a group composed of the object *Counselor*.

The chosen granularity is the file level, but for two User stories editing the same file it could be interesting to go deeper in the source code itself. To identify more variability, we can create clusters of similar User stories. For example we can connect the first User story with the fourth, because both shared a plot visualisation.

Assisting the feature model synthesis is supported by the variability extracted from the User stories. The Feature model is completed by the relations resulting from RCA. Each feature has its group of associated code files. From these two parts it is more feasible to derive new products from feature selection. In our example, the current product could be regenerated from the expressed features. We aim to be able to reproduce the existing products from the obtained result.

5 WORK PLAN

In the context of the PhD thesis, we have already developed a method based on FCA to extract variability from Input and Output data schemata of simulators in Agriculture Decision-Support Software [8]. This method provided interesting preliminary results. It needs to be extended by extracting variability from other software artifacts. This is why we are studying the analysis of User stories and Merges in this work.

Besides this, and at the very beginning of the thesis, we built and conducted interviews to evaluate the perception of the migration by the employees of our industrial partner. We followed a protocol to create a questionnaire and to conduct interviews. Then we analyzed the data collected from these interviews. We wrote a paper and submitted it to an international conference. Our submission is under review.

Already one and a half years were dedicated to completing these two works and still another year and a half until the end of the PhD thesis which will be dedicated to the work presented in this paper.

For the next 12 months, we plan to pursue the work presented in this paper. We will start working on a Proof Of Concept on a relatively small data-set from the company (300 user stories and 500 Merge-requests among 4000 Merge-requests in total, related to all stories), in order to evaluate the feasibility and identify the limits of our method.

Finally, we plan to fine-tune and evaluate our method on larger datasets. This will enable us to evaluate the generalization of this method, by testing it on public projects hosted on Collaborative version control platform as GitHub, GitLab or Jira. Implementing a reusable tool to explore source code variability extracted from User stories is included in this step.

Some publications are planed in this thesis. First we will work on a contribution on feature/variability identification and location, which was introduced in this paper. Then a second publication will be written on interview reviews after the migration to assess developer feedback.

Then the last six months will be devoted to write the PhD Thesis and finally the PhD oral examination.

6 CONCLUSION

This paper presents an approach to support Software Product Line Migration by integrating agile specifications expressed by the User stories. This approach includes the design of an approach to identify then locate features from Collaborative version control platforms. Feature variability is identified from User stories for feature models synthesis. Then feature models are enriched by analyzing links between source code and features.

This approach uses Relational Concept Analysis and Formal Concept Analysis to group and show correspondences between the agile specifications and the source code. This approach aims to highlight features and variability between features identified in the User stories and their relations to the Merges.

We will evaluate our approach. First we will conduct an experiment on a small set of data to study its feasibility and adjust the methodology. Next we will work on data from our industrial partner to assess its applicability on a real-world professional setting. Finally we will consider data from public projects to assess the generalization of the approach.

REFERENCES

[1] Thorsten Berger, Daniela Lettner, Julia Rubin, Paul Grünbacher, Adeline Silva, Martin Becker, Marsha Chechik, and Krzysztof Czarnecki. 2015. What is a Feature? A Qualitative Study of Features in Industrial Software Product Lines. In *Proceedings of the 19th International Conference on Software Product Line* (Nashville, Tennessee) *(SPLC '15)*. Association for Computing Machinery, New York, NY, USA, 16–25. https://doi.org/10.1145/2791060.2791108

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[3] Jessie Carbonnel, Marianne Huchard, and Alain Gutierrez. 2015. Variability representation in product lines using concept lattices: feasibility study with descriptions from Wikipedia's product comparison matrices. In *FCA&A-ICFCA: International Conference on Formal Concept Analysis - International Conference on Formal Concept Analysis*, Manuel Ojeda-Aciego, Jaume Baixeries, and Christian Sacarea (Eds.), Vol. CEUR Workshop Proceedings. University of Málaga, Nerja, Málaga, Spain. https://hal-lirmm.ccsd.cnrs.fr/lirmm-01183447

[4] Jessie Carbonnel, Marianne Huchard, André Miralles, and Clémentine Nebut. 2017. Feature Model Composition Assisted by Formal Concept Analysis. In *ENASE: Evaluation of Novel Approaches to Software Engineering (Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering)*, Ernesto Damiani, George Spanoudakis, and Leszek A. Maciaszek (Eds.). SciTePress, Porto, Portugal, 27–37. https://doi.org/10.5220/0006276600270037

[5] 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

[6] S.T. Dumais. 2004. Latent Semantic Analysis. *Annual Review of Information Science and Technology* 38 (01 2004), 188–230. https://doi.org/10.1002/aris.1440380105

[7] Bernhard Ganter and Rudolf Wille. 2012. *Formal concept analysis: mathematical foundations.* Springer Science & Business Media.

[8] Thomas Georges, Marianne Huchard, Mélanie König, Clémentine Nebut, and Chouki Tibermacine. 2021. Variability Extraction from Simulator I/O Data Schemata in Agriculture Decision-Support Software. In *Proceedings of the 9th Workshop (FCA4AI), co-located with IJCAI 2021, Montréal, Canada, August 2021. Virutal Event.*

[9] Marianne Huchard. 2016. Relational Concept Analysis: An approach for classifying and mining multi-relational data. https://hal.archives-ouvertes.fr/hal-01316104 Invited seminar at NorthEastern University, State Key Laboratory of Synthetical Automation for Process Industries, Shenyang.

[10] Lukas Linsbauer, E. Roberto Lopez-Herrejon, and Alexander Egyed. 2013. Recovering Traceability between Features and Code in Product Variants. In *Proceedings of the 17th International Software Product Line Conference* (Tokyo, Japan) *(SPLC '13)*. Association for Computing Machinery, New York, NY, USA, 131–140. https://doi.org/10.1145/2491627.2491630

[11] 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

[12] Elallaoui Meryem, Khalid Nafil, and Raja Touahni. 2018. Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques. *Procedia Computer Science* 130 (01 2018), 42–49. https://doi.org/10.1016/j.procs.2018.04.010

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[14] Hlad Nicolas, Bérénice Lemoine, Marianne Huchard, and Abdelhak-Djamel Seriai. 2021. Leveraging relational concept analysis for automated feature location in software product lines. In *GPCE 2021 - 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*. Association for Computing Machinery (ACM SIGPLAN), Chicago, United States, 170–183. https://doi.org/10.1145/3486609.3487208

[15] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. 1998. Latent Semantic Indexing: A Probabilistic Analysis. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Seattle, Washington, USA) *(PODS '98)*. Association for Computing Machinery, New York, NY, USA, 159–168. https://doi.org/10.1145/275487.275505

[16] Daniel Pech, Jens Knodel, Ralf Carbon, Clemens Schitter, and Dirk Hein. 2009. Variability management in small development organizations: experiences and lessons learned from a case study. 285–294. https://doi.org/10.1145/1753235.1753274

[17] Klaus Pohl, Günter Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering - Foundations, Principles, and Techniques.* Springer. https://doi.org/10.1007/3-540-28901-1