Exploring the 3-dimensional variability of websites' user-stories using triadic concept analysis

Alexandre Bazin, Thomas Georges, Marianne Huchard, Pierre Martin, Chouki Tibermacine

International Journal of Approximate Reasoning 173 (2024) 109248

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Exploring the 3-dimensional variability of websites' user-stories using triadic concept analysis

Alexandre Bazin[a,*], Thomas Georges[a,b], Marianne Huchard[a], Pierre Martin[c], Chouki Tibermacine[a]

[a] *LIRMM, Univ Montpellier, CNRS, Montpellier, France*
[b] *ITK -Predict & Decide, Montpellier, France*
[c] *AIDA, Cirad, Montpellier, France*

ARTICLE INFO

Keywords:
Variability modeling
Software product line
Agile development
User story
Formal concept analysis
Triadic concept analysis
Implication rule

ABSTRACT

Configurable software systems and families of similar software systems are increasingly being considered by industry to provide software tailored to each customer's needs. Their development requires managing software variability, i.e. commonalities, differences and constraints. A primary step is properly analyzing the variability of software, which can be done at various levels, from specification to deployment. In this paper, we focus on the software variability expressed through user-stories, viz. short formatted sentences indicating which user role can perform which action at the specification level. At this level, variability is usually analyzed in a two dimension view, i.e. software described by features, and considering the roles apart. The novelty of this work is to model the three dimensions of the variability (i.e. software, roles, features) and explore it using Triadic Concept Analysis (TCA), an extension of Formal Concept Analysis. The variability exploration is based on the extraction of 3-dimensional implication rules. The adopted methodology is applied to a case study made of 65 commercial web sites in four domains, i.e. manga, martial arts sports equipment, board games including trading cards, and video-games. This work highlights the diversity of information provided by such methodology to draw directions for the development of a new product or for building software variability models.

1. Introduction

Configurable software (e.g. Linux system) and families of similar software (e.g. a family of e-commerce websites in a specific domain) have become common. Ensuring the efficient design, development, deployment and maintenance of such software requires managing their many variants. This is addressed in software engineering using various strategies, such as the ad-hoc Clone-and-Own principle, Software Product Line Engineering (SPLE), or combining both [1]. In SPLE, variability is managed through a unique common platform including variability models and software assets, and tools allowing to derive software variants in a disciplined way [2,3]. The platform capitalizes on best practices in software development and concrete reusable artifacts applied to a considered domain in order to provide a product that best meets users' need. It also reduces the production cost and increases the software quality. As a first step in software development, models and their variability are designed at diverse abstraction levels and expressed

* Corresponding author.
E-mail address: alexandre.bazin@lirmm.fr (A. Bazin).

https://doi.org/10.1016/j.ijar.2024.109248
Received 1 March 2024; Received in revised form 12 May 2024; Accepted 3 July 2024
Available online 14 July 2024

using a formalized language. One of the challenges is thus to develop a method that improves this modeling task and reduces its duration. This involves modeling and representing commonalities, differences, and constraints.

In this paper, we model variability at the specification step. In companies that follow the *Agile approach* principles [4], a widespread practice to capture requirements is to write user-stories. These are short formatted sentences indicating the action (or more generally the accessible feature) and purpose performed by a user role, the purpose generally being omitted. We can reasonably consider that the user-story sets of a family of similar systems are representative of the family variability at the specification step. Analyzing this information can help software developers as a decision support to develop a major part of the variability models. It can also serve to take decisions when a new software variant is designed and implemented.

This paper aims to develop a methodology to support such an analysis. Compared to the classical variability analyses based on a 2-dimensional framework, i.e. software systems and features, analyzing user-stories requires to consider 3 dimensions, i.e. software systems, roles, and features. We did not find approaches in the software variability literature that consider these 3 dimensions in the variability analysis. Inspired by previous variability analysis methods that consider dimensions and rely on formal concept analysis (FCA) [5–7], we thus explored a new approach based on triadic concept analysis (TCA), an extension of FCA to 3-dimensional datasets, which captures variability information as a whole.

The adopted methodology relies on extracting 3-dimensional implication rules and giving clues to interpret them in terms that correspond to the concerns of a software engineer. In this paper, we apply our methodology on a significant set of user-story sets provided by supervised Bachelor students from 65 commercial web sites from the user interface, completed by guessed internal features. The web sites come from four domains: mangas and derived products, martial art equipment, board games and video games. One co-author of this paper reviewed and carefully fixed and standardized these user-stories. We then evaluate the applicability, scalability, and the value of the identified variability. We conclude both on feasibility and relevancy of the approach.

The remainder of this paper is as follows. Section 2 motivates the need for extracting 3-dimensional variability relationships as a complement of the 2-dimensional ones. A short example inspired by the case study, dealing with *Manga online stores*, illustrates this motivation. Section 3 introduces triadic concept analysis (TCA) [8], i.e. the adopted method to extract this particular kind of variability relationships. Section 4 describes the case study, and Section 5 the related work. Finally, Section 6 concludes this paper, and gives a few perspectives of this work.

2. Motivation

Expressing software requirements took different forms over the years, that went from textual use-case descriptions to the use-case diagrams of UML. Nowadays, a very common practice in industry consists to write the requirements through simple user-stories. These are short sentences aimed at describing a behavior of the system that has some value for a user. They are defined along the template *As a < user >, I want to < action >, so that < benefit >*, where an action corresponds to a feature (also known as a functionality). But, they are often reduced in practice to *As a < user >, I want to < action >*, as reported in [9]. In a family of similar software, each software system has its own set of user-stories, named a family of user-story sets. Table 1 presents a fictive example, named *Manga online stores*, of user-story sets from 4 similar software systems dedicated to the sale of mangas. Exploring the variability in these user-story sets deserves attention, as software engineers could exploit it to build assets useful for recommending and deriving new products.

Currently, in software product lines, variability analysis considers systems described by features, thus ignoring the dimension of the roles that user-stories highlight. As a result, variability is reduced to a 2-dimensional representation space (systems and features). The variability, that interests software engineers, involves logical relationships between features, such as binary implications, mutual exclusions, or groups (or/xor). For instance, in Table 1, one can observe the two following logical relationships: $addComment$ and $viewComment$ always appear together; if a system has the feature $manageCart$, it also has feature pay. From these logical relationships, engineers synthesize *variability models*, which can take several forms, going from CSP (Constraint Satisfaction Problems) [10] to graphical models such as *feature models*, which are kinds of logical trees capturing the variability constraints as a whole and representing all possible system configurations [11]. Feature model synthesis is a core topic in SPL, combining logical and ontological information [12]. These feature models are then used in practice to create tools that assist users in the configuration of new products. Typical examples are vehicles configurators in automotive industry. In the case of software, once the configuration has been achieved, the code of a nearly executable application is generated. The variability analysis is thus a crucial step, initiating the process of building the product line.

Extending variability analysis to 3 dimensions, i.e. systems × roles × features would be very beneficial in this engineering because ignoring the role dimension discards some relevant information. For instance, all the systems share the feature $CRUDproducts$ (Create Read Update Delete products), but it is handled by different roles: by $productOwner$ in $MyManga$ and $MangaStore$, and by $admin$ in $MangaFan$ and $MangaHome$. There is thus an interest in considering more dimensions in the variability analysis to produce more precise feature models and thus better configurations and generated software. One could remark that, even though this is not classically done in the SPL field, it is possible to consider roles in 2-dimensional analyses, e.g. by considering which roles appear in which system, or which role can achieve which feature. E.g., in systems × roles perspective, it can be observed that in all systems, there is the role user, or if there is a registered user in a system, there is also an admin. But this does not capture the whole variability, as it can be noticed that each role may access different features depending on the systems. Using a single perspective based on 3-dimensional descriptions, additional subtle logical relationships can indeed be identified: (1) between roles, based on roles shared by existing pairs (system, feature); (2) between features, based on existing pairs (system, roles), (3) between pairs (role, feature), based

Table 1
The fictive example *Manga online stores* composed of the user-story sets from the four software `MyManga`, `MangaStore`, `MangaFan`, and `MangaHome`.

System	Role	Feature
MyManga	user	search
MyManga	user	viewComment
MyManga	user	addComment
MyManga	productOwner	CRUDproducts
MyManga	communityManager	moderateComment
MangaFan	user	search
MangaFan	user	manageCart
MangaFan	user	pay
MangaFan	user	signUp
MangaFan	admin	CRUDproducts
MangaFan	admin	manageUserAccount
MangaFan	registeredUser	manageProfile
MangaFan	admin	manageProfile

System	Role	Feature
MangaStore	user	search
MangaStore	user	manageCart
MangaStore	user	pay
MangaStore	productOwner	makePromotionalCampaign
MangaStore	productOwner	CRUDproducts
MangaHome	user	search
MangaHome	user	viewComment
MangaHome	user	addComment
MangaHome	communityManager	moderateComment
MangaHome	user	signUp
MangaHome	admin	CRUDproducts
MangaHome	admin	manageUserAccount
MangaHome	registeredUser	manageProfile
MangaHome	admin	manageProfile

Table 2
A formal context depicting the relation between software systems and their features.

	search	viewComment	moderateComment	manageCart
MyManga	×	×	×	
MangaStore	×			×
MangaFan	×			×
MangaHome	×	×	×	

on systems, and (4) various kinds of conditional relationships. For instance, in Table 1, the following logical relationships can be extracted:

- If registered users handle a feature in a system, then admins handle it as well (e.g. $manageProfile$). Roles $\{registeredusers, admins\}$ are indeed shared by pairs $(MangaFan, manageProfile)$ and $(MangaHome, manageProfile)$.
- If a role can manage user profiles and CRUD products in a system, then it can also manage user accounts. Indeed, when features $\{manageProfile, CRUDproduct\}$ are available for a pair (System, role), e.g. $(MangaFan, admin)$ or $(MangaHome, admin)$, then the feature $manageUserAccount$ is also available for this pair.
- If a system, i.e. $MyManga$ or $MangaHome$, allows community managers to moderate comments ($communityManager$, $moderateComments$), then it also allows users to add comments ($user, addComments$), and reciprocally.
- If the role $admin$ and the feature $CRUDproducts$ are present, then the feature $manageUserAccount$ is also present. The relation between these features is not true for $MyManga$ or $MangaStore$ which have $CRUDproducts$, but neither role $admin$, nor feature $manageUserAccount$.

All these relationships can lead to variability models including roles and features such as variable use case diagrams [13], guiding product configuration, making recommendations for new products based on most common configurations, or detecting rare or incorrect configurations. In this paper, we propose a new method to explore their extraction using triadic concept analysis (TCA), an extension of formal concept analysis (also called dyadic concept analysis), which is presented in the next section.

3. Dyadic and triadic concept analysis

This section introduces formal (dyadic) concept analysis (FCA) and its tridimensional generalisation, viz. triadic concept analysis (TCA).

Formal concept analysis (FCA), also called dyadic concept analysis, is a mathematical framework based on lattice theory that aims at structuring the information contained in the relation between *objects* and their *attributes* [14]. This structuring takes the form of *concepts*, i.e. groupings of objects sharing the same attributes, or *implications*, i.e. regularities in the description of objects by attributes. In this paper, we are interested in implications.

Binary relations are called formal contexts and are formalised as a triple $(\mathcal{O}, \mathcal{A}, \mathcal{R})$ where $\mathcal{O}$ is a set of objects, $\mathcal{A}$ is a set of attributes and $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ is an incidence relation between objects and attributes. We say that *the object o is described by the attribute a* when $(o, a) \in \mathcal{R}$. Formal contexts are usually represented as binary tables such as Table 2 depicting the relation between software systems (objects) and their features (attributes).

Implication rules are regularities of the form $A \rightarrow B$, where A and B are sets of attributes, meaning that all objects described by the attributes in A are also described by the attributes in B. We name *support* of the rule the set of objects described by A (and, thus, by $A \cup B$). Sometimes, when it does not introduce ambiguity, we also use *support* to designate the cardinality of the support. For instance, in Table 2, the following implications may be observed:

Table 3
A triadic context $(\{MyManga, MangaStore, MangaHome\}, \{search, viewComment, manageCart\}, \{FinalUser, Administrator, ProductManager\}, \mathcal{R})$.

	$search$	$viewComment$	$manageCart$	$search$	$viewComment$	$manageCart$	$search$	$viewComment$	$manageCart$
$MyManga$	×			×		×			×
$MangaStore$	×			×					
$MangaHome$	×	×			×			×	×
	$FinalUser$			$Administrator$			$ProductManager$		

Table 4
The contexts $C^{(\{S,F\},\{R\})} = C^{(\{Systems,Features\},\{Role\})}$ (bipartitioning) and $C_{\{FinalUser,Administrator\}}$ (conditioning), where C is the Table 3 tricontext.

	$FinalUser$	$Administrator$	$ProductManager$
$(MyManga, search)$	×	×	
$(MyManga, viewComment)$			
$(MyManga, manageCart)$		×	×
$(MangaStore, search)$	×	×	
$(MangaStore, viewComment)$			
$(MangaStore, manageCart)$			
$(MangaHome, search)$	×		
$(MangaHome, viewComment)$	×	×	×
$(MangaHome, manageCart)$			×

	$search$	$viewComment$	$manageCart$
$MyManga$	×		
$MangaStore$	×		
$MangaHome$		×	

$$\emptyset \to \{search\}$$
$$\{viewComment\} \to \{moderateComment, search\}$$
$$\{moderateComment, search\} \to \{viewComment\}$$
$$\{viewComment, moderateComment\} \to \{search\}$$

The first implication, $\emptyset \to \{search\}$, means that all software systems offer the search feature. The set of all the implications in this formal context is clearly redundant, as $\{viewComment, moderateComment\} \to \{search\}$ can be inferred from $\emptyset \to \{search\}$. Thus, one is often interested in subsets of implications called *implication bases* that minimize redundancy while still retaining all the information. One such implication base is made of the implications of the form $A \to \{b\}$ where A is a cardinality-minimal set of attributes such that the rule holds called a *proper premise* [15]. The proper premises implication base (PPIB) of Table 2 context is:

$$\emptyset \to \{search\}$$
$$\{viewComment\} \to \{moderateComment\}$$
$$\{moderateComment\} \to \{viewComment\}$$

Alternatively, one could consider implications between sets of objects as the two interchangeable dimensions. However, these are usually less useful.

TCA [8], a natural extension of FCA, is aimed at ternary relations. A *tricontext* is a quadruple $(\mathcal{O}, \mathcal{A}, \mathcal{C}, \mathcal{R})$ in which $\mathcal{O}$ is a set of *objects*, $\mathcal{A}$ is a set of *attributes*, $\mathcal{C}$ is a set of *conditions* and $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A} \times \mathcal{C}$ is a ternary relation between objects, attributes and conditions. When $(o, a, c) \in \mathcal{R}$, we say that *the object o is described by the attribute a under the condition c*. Table 3 depicts a tricontext describing *systems* offering different *features* to different *roles*.

Just as in the dyadic case where implications between objects or between attributes exist, the triadic case offers different types of triadic implications [16,17]. To produce such triadic implications, two operations on the dimensions of the tricontext C have to be applied: partitioning and conditioning. Partitioning consists in creating a dyadic formal context by bipartitioning the set of dimensions, the Cartesian product of each partition producing a dimension of the new context. Given a bipartition Ω, we call C^{Ω} such a dyadic context resulting from the bipartition of the dimensions of the tricontext C. Table 4 (left-hand side) depicts the example for the bipartition $\{\{Systems, Functionalities\}\{Role\}\}$. Conditioning consists in restricting the incidence relation of the tricontext to a subset of a dimension by only keeping the crosses that exist for each element of this subset, effectively removing the conditioning dimension. Given a subset X of a dimension, we call C_X the dyadic context resulting from the conditioning of the tricontext C by X. Table 4 (right-hand side) depicts an example of conditioning, restricting to the subset $\{FinalUser, Administrator\}$.

As partitioning and conditioning give rise to dyadic formal contexts, they also give rise to the corresponding implication bases. As such there are six types of non-conditioned implications in a triadic context: rules between objects, between attributes, between conditions, between pairs (object,attribute), between pairs (object,condition) and between pairs (attribute,condition). Additionally, there are $2^{|\mathcal{O}|} - 1$ bases of implications conditioned with all the subsets of objects, $2^{|\mathcal{A}|} - 1$ bases of implications conditioned with all the subsets of attributes and $2^{|\mathcal{C}|} - 1$ bases of implications conditioned with all the subsets of conditions.

For instance, the implication $\{FinalUser, ProductManager\} \to \{Administrator\}$ holds in the Table 3 tricontext C because it holds in the $C^{(\{S,F\},\{R\})}$ context (depicted in Table 4), i.e. all pairs $(system, feature)$ that appear in the incidence relation with both $FinalUser$ and $ProductManager$ also appear with $Administrator$. Those pairs are limited to $\{(MangaHome, viewComment)\}$. Similarly, the implication $\{(viewComment, Administrator\} \to \{(manageCart, ProductManager)\}$ between pairs $(feature, role)$ holds in C because it holds in $C^{(\{Products\},\{Features,Roles\})}$. Its support is $\{MangaHome\}$. The conditioned implication $(\{viewComment\} \to \{search\})_{\{FinalUser\}}$ also holds as it holds in the $C_{\{FinalUser\}}$ context. Its support is then $\{MangaHome\}$.

4. Case study

This section presents the case study. Section 4.1 introduces the explored questions and the dataset. Section 4.2 presents the experimental setup, the tool, and the metrics chosen for answering the questions. Section 4.3 shows the results per metrics. Section 4.4 uses the metric results for answering the questions, and discusses the threats to validity.

4.1. Method

Explored questions In the case study, we explored three main questions:

- **Q1. Applicability**. A user-story intrinsically describes requirements along several dimensions. In this work, we considered the three dimensions: system, role, and feature. Another applicability question is related to the heterogeneity of points of views of the various teams. We notice differences in the granularity of the descriptions, and in the terms adopted to formulate a feature and a role. We applied a thorough process of alignment on the user-story sets produced by the students. We will evaluate this process.
- **Q2. Scalability**. FCA and TCA are "exact data mining methods". By exact, we mean that the mined patterns are characterized and all patterns are extracted. E.g. Implications bases exhaustively report and summarize the logical information from the dataset. This comes with a potentially exponential running time and result size. "Pattern deluge" is a common question in data mining. We will consider this aspect as a second question.
- **Q3. Value of the extracted patterns**. We will assess whether a sufficient number of logical relationships, with a significant generality, were extracted, and whether interesting things were learned.

Dataset building We asked 175 Bachelor students, in their second year of computer science studies, to group by two or three and then to select one existing sale/purchase website from a list of four topics: mangas and derived products, sport equipment for martial arts, board games, and video games. Each group had to browse the website to infer the available features and write the corresponding user-stories using the template (*As a, I want to, so that*). To conduct this work, groups could consult online documentation and could use chatGPT to conceive features on the non accessible website part, but providing their source of information in their final report. They also had to draw an UML use case diagram, consistent with the user-stories. This part of the exercise aimed to make them think more deeply about the requirements of the user-stories to improve their quality. Discussion between groups was allowed. The work was mainly carried out in classrooms and supervised by four different teachers allowed to provide assistance to the students. From this work, we collected user-stories for 65 websites and 67 website descriptions, as 2 websites were described by 2 groups each.

A co-author of this paper then transformed the user-story descriptions, provided as literal text, into normalized and structured data to enable their analysis. The structure of the description of a user-story differentiated the role from the feature, i.e. the role was defined as a subject (e.g. user, administrator) and the feature was handled as an action composed of the four elements V, O, A, and S, corresponding respectively to a transitive action verb (e.g. to search), a direct object of the verb (e.g. a resource such as manga), an attribute of the action such as a manner (e.g. using keywords), and the support of the action (e.g. book list of the website). The transcription of the literal text into the role and each V, O, A, and S element was conducted manually. The normalization of each part of the structure was then conducted manually using an iterative process based on the construction of vocabularies, considering roles and each of the four elements V, O, A, and S. The aim of the vocabularies was to group the terms in a lexicon and align each one with a literal concept that merges similar values. The alignment was first done by group and then for all the groups. For instance, the literal concept "Final user", adopted in the final dataset to describe a role, regroups the terms "User", "Final customer", "buyer", "guest", "new user", "logged user", "visitor", and "registered user". This process was inspired by OpenRefine [18]. Finally, five iterations produced the final dataset,[1] made of 1546 user-stories, once duplicates were removed. Vocabularies of role, V, O, A, and S contained 17, 30, 59, 78, and 10 literal concepts respectively.

4.2. Experimental setup

Used parts of the dataset For the evaluation, we considered the 67 systems, with their 17 roles, and 1546 user-stories. In what follows, the direct object of the action verb, named O in the description structure, was renamed Resource, to avoid confusion with the objects of a formal context. Two kinds of triple construction have been considered:

- $(system, role, action\ verb)$. In these triples, only the action verb is used to describe the feature, leading to 30 features.
- $(system, role, action\ verb + resource)$. The feature is described by the action verb concatenated to the resource, giving potentially more variation, but a more precise meaning as a counterpart, leading to 145 features.

[1] https://doi.org/10.18167/DVN1/BWCC71.

Table 5
Statistics on the numbers of rules for different values of supports.

	ActionVerb					ActionVerbResource				
	Features	Roles	Feature× Roles	Role-cond	Feat-cond	Features	Roles	Feature× Roles	Role-cond	Feat-cond
#support 0	8881	1530	588056	7113	1957	1432743	1652	5449147	209628	622
#support 1	223	38	22663	355	193	8117	34	111644	6833	34
support max	47	5	50	49	10	47	1	48	48	6
#support max	1	2	1	1	1	1	34	1	1	1

Tool Proper premises, and thus implications, can be computed through the iteration of an algorithm for computing the minimal transversals of a hypergraph. We implemented the algorithm from [15][2] in Python 3.10 using Takeaki Uno's SHD software[3] [19] to compute minimal transversals.

Chosen metrics In order to answer our three questions, we computed the following metrics: Metrics on the alignment (Duration of the manual work, Number of initial roles and aligned roles, Number of initial verbs and aligned verbs, Number of initial resources and aligned resources); Running time; Implication number presented per support, for each observed dimension; Implication number per conclusion, for each observed dimension. Alignment metrics serve to answer to **Q1 (Applicability)**. Metrics on running time and implication number provide answers to **Q2 (Scalability)**. Metrics on implication number per support and per conclusion contribute to answer question **Q3 (Value of the extracted patterns)**.

4.3. Results

Metrics on alignment Thirty five hours were required to align the terms adopted to normalize and structure the description of the user-stories. From the 97 initial roles provided by the student groups, 17 were finally identified. This strong reduction is due to the presence of many synonyms (e.g. 30 terms were variations of the user's name) and mistyping terms among others. The transcription of the user-study features into the V, O, A, S structure provided respectively, 37 verbs, 78 resources, 58 attributes, and 10 supports. Once the terms were aligned, the final number of concepts were 30, 59, 78, and 10 respectively. The reduction of the number of verbs and resources corresponded mostly to the grouping of synonyms. The increase in the number of attributes (i.e. A in the description structure) resulted mainly from importing terms from the O structure when aligning the complete descriptions V, O, A, S with each other. Finally, the supports remained unchanged.

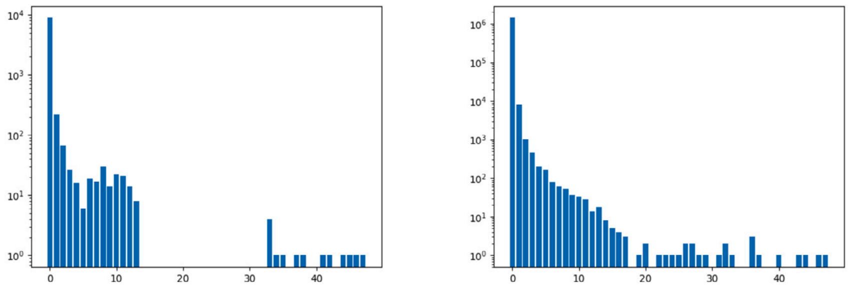
Running time The experiments were performed on a i7-1165G7 2.80 GHz CPU with 16 GB RAM. When features are reduced to action verbs, it took respectively 144 sec (about 2 min), 418 sec (about 7 min) and 3615 sec (about 1 h) to compute the implications for roles, for features, and for pairs $(role, feature)$ in the whole dataset. When resources are concatenated to action verbs, the running time is drastically increased, being respectively 602 sec (about 10 min), 58020 sec (about 16 h) and 32829 sec (about 9 h). Conditional implications are much faster to compute as they involve one less dimension, with running time 287 sec (below 5 min) for role-conditioned implications when resources are concatenated to action verbs, 2 sec when only action verbs are considered. For feature-conditioned implications, in both cases, it is 0.5 sec.

Implication number per support Figs. 1a, 1b, and 1c present the number of implication rules between, respectively, features, roles, and pairs (role,feature) for each support within the two excerpts of the dataset. Figs. 2a and 2b present the same nature of information for, respectively, role-conditioned and feature-conditioned implications. The y-axis uses a logarithmic scale. We observe that rules with a support equal to 0, corresponding to mutual exclusion, are the most numerous. For instance, the ActionVerbResource excerpt contains 5449147 rules between pairs (role,feature) but "only" 127891 rules with a support ≥ 1. Most of the rules have a relatively low support, meaning that some of them can safely be considered quirks of the excerpt instead of relevant information on the regularities. However, some rules do have significant support and should be considered relevant. For instance, the ActionVerb excerpt has 12 implications between pairs (role,feature) with a support greater than 30 ($\sim$44% of the systems) and 14 rules between features with a support greater than 32 ($\sim$2% of the pairs (system,role)).

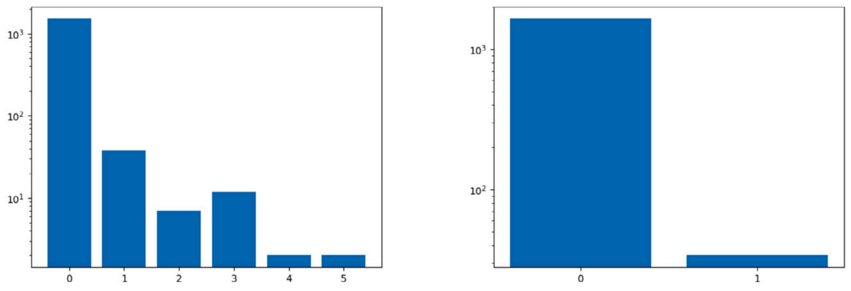
We also observe that the number of implications per subset differs from one subset to another, with rules between pairs (role,feature) being the most numerous. For instance, the ActionVerb excerpt has respectively a total of 498, 61, and 27753 rules with a support ≥ 1 between features, between roles, and between pairs (role,feature). This result is not surprising as the number of pairs is the product of the number of roles and the number of features. In addition, it is known that the number of implications grows exponentially with the number of attributes [20]. A summary is given in Table 5.

Implication number per conclusion Figs. 3a, 3b, and 3c depict the number of rules in which each "attribute" ($role$, $feat$ or pair $(role, feat)$, respectively) is located in the conclusion. Figs. 4a and 4b depict the same information for, respectively, role-conditioned and feature-conditioned implications. Attributes have been ordered by decreasing number of rules. The y-axis uses a logarithmic scale. In the ActionVerb excerpt, the most frequent feature in the conclusion part of the rules is *search* observed in 68 rules. The most

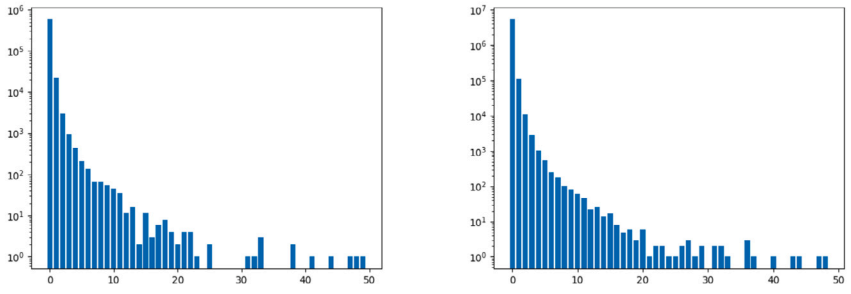
[2] https://github.com/Authary/PCA.
[3] http://research.nii.ac.jp/~uno/dualization.html.



(a) Number of rules per support between features: ActionVerb (left-hand side), ActionVerbResource (right-hand side).



(b) Number of rules per support between roles: ActionVerb (left-hand side), ActionVerbResource (right-hand side).



(c) Number of rules per support between pairs (role,feature): ActionVerb (left-hand side), ActionVerbResource (right-hand side).

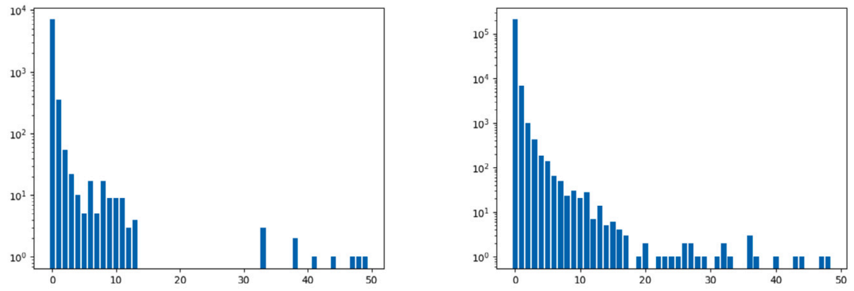
Fig. 1. Number of rules per support. The y-axis uses a logarithmic scale.

frequent conclusion in implications between pairs (role,feature) is *(Final user,contact)*, observed in 796 rules. We also observed that some attributes are more frequent than others in the conclusions of the implications. In the ActionVerb excerpt, 13% of the features, 12% of the roles, and 4% of the pairs (role,feature) make up the conclusions of 50% of the rules, and 35% of the features, 47% of the roles and 10% of the pairs (role,feature) make up the conclusions of 90% of the rules.

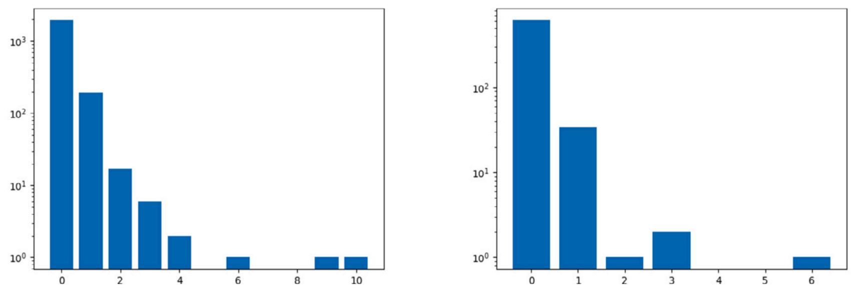
4.4. Discussion

4.4.1. Answering the questions

Q1. Applicability The duration for conducting the manual alignment of the user-stories in the dataset can be considered large. We decided to proceed manually for several reasons: (1) to guarantee a certain quality on the dataset, as the students are beginners in requirement analysis; (2) to evaluate the time such work could take, and (3) to dispose of a ground truth for further experiments. The differences between the initial and the final vocabulary on roles and features, unsurprisingly show that the method is applicable only if such a step is properly performed, either manually or automatically. The strongest difference appears for roles. Apart from the numerous variations on the final user denomination for which we do not have the explanation, this variability can be due to the fact that students had to guess the names of the roles, corresponding to the back-end management, by navigating the front-end,



(a) Number of rules per support between features conditioned with role sets: ActionVerb (left-hand side), ActionVerbResource (right-hand side).



(b) Number of rules per support between roles conditioned with feature sets: ActionVerb (left-hand side), ActionVerbResource (right-hand side).

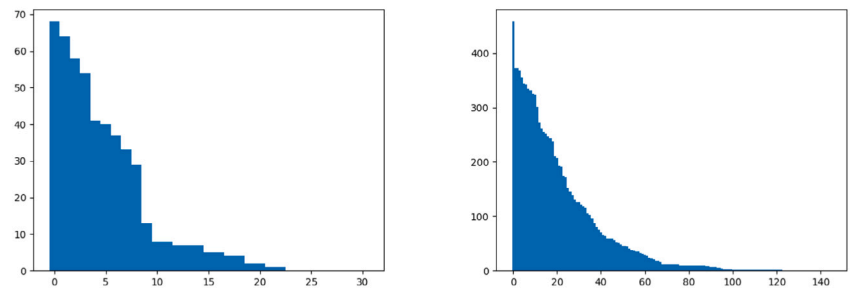
Fig. 2. Number of conditional rules per support. The y-axis uses a logarithmic scale.

and using mainly their own experience. The difference in variability between the V, O, A, S vocabularies may depend on the English distribution of terms in substantives and verbs and on the skills of the students, a phenomenon that is difficult to assess. In the near future, we plan to compare different automatic or semi-automatic methods to achieve this alignment, and having this ground truth will be useful. We also will consider extracting more dimensions, e.g. considering separately the action verb, the resource (i.e. the direct object), and adding the attribute of the action.

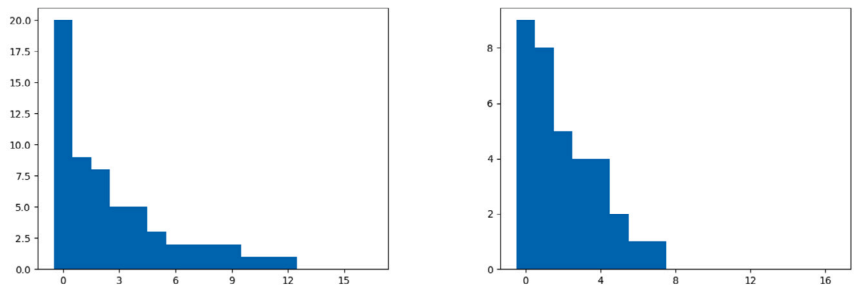
Q2. Scalability The results show the feasibility in running time, but also that the implementation deserves to be optimized. However, as the approach is designed to be part of an analysis process, the implications can be computed off-line and analyzed later. When action verbs and resources are concatenated, the running time is not surprisingly higher due to the increase of the number and variability of the data. This may explain the large amount of time (16 h) needed to compute the implications for features. An additional analysis will allow to verify this hypothesis.

The result size, in particular the numbers of implications, show that there are many implications with a support equal to 0. These implications may be useful as they indicate mutual exclusion. However, it will be hard to use them without a post-processing task. We can suspect that many of them are accidental. Furthermore, the implications with a low support need also be post-processed. E.g. implications with support equal to 1 are specific to only one system, and can be discarded. For other low supports, it depends on the purpose of the analysis, as a low support means a rare pattern, and may be of interest for analysts who want to find niche products.

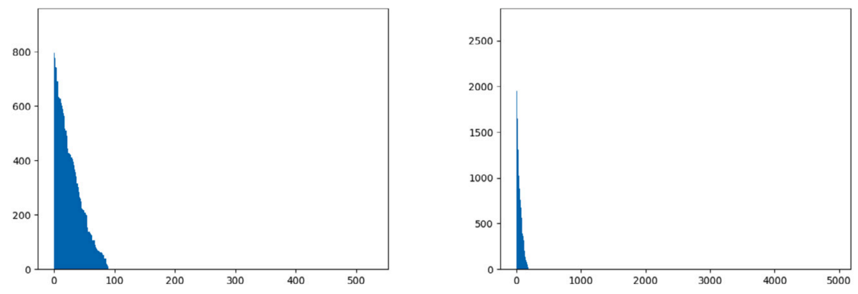
Q3. Value of the extracted patterns The extracted patterns are valuable if they give relevant information to build a variability model. It could be argued that a set of implications is per se a variability model, and it is fair if this model has to be mainly managed by tools. But software engineers usually look at their models, especially when they come from a reverse engineering process, as in our case, to fix or improve it. Thus synthesizing a graphical variability model from logical relationships (often restricted to co-occurrences, binary implications, and binary mutual exclusions) completed by ontological information is common, as shown by [12,21] for the dyadic case. There are very few proposals in the SPLE domain to graphically represent 3-dimensional variability with systems, roles and features. Currently, the most advanced representation is the variable use case diagram introduced by [13]. This diagram enables representing mandatory and optional roles, uses cases, and relations between use cases. It also admits auxiliary textual constraints. It remains to be proven that these diagrams enable representing all the complex information coming from triadic implications. Whatever it is, the process of going from an implication set to the graphical representation requires the expert not to be overwhelmed by information and the implications having informative content. As shown by the figures, there is a reasonable number of implications, with a high support, that can be analyzed. Filtering implications by support may be a guiding analysis scheme. Other techniques may help to make a systematic analysis without being overwhelmed, such as sorting rules by their content, e.g. for analyzing all implications that



(a) Number of rules between features, each feature being in the conclusion of: ActionVerb (left-hand side), ActionVerbResource (right-hand side). Features have been ordered by decreasing number of rules.



(b) Number of rules between roles, each role being in the conclusion of: ActionVerb (left-hand side), ActionVerbResource (right-hand side). Roles have been ordered by decreasing number of rules.

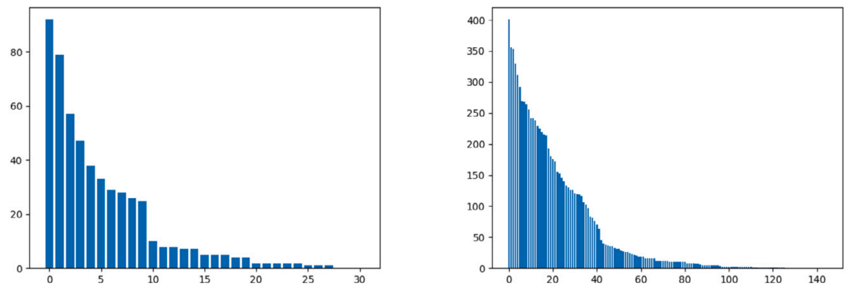


(c) Number of rules between pairs (role,feature), each pair (role,feature) being in the conclusion of: ActionVerb (left-hand side), ActionVerbResource (right-hand side). Pairs (role,feature) have been ordered by decreasing number of rules.

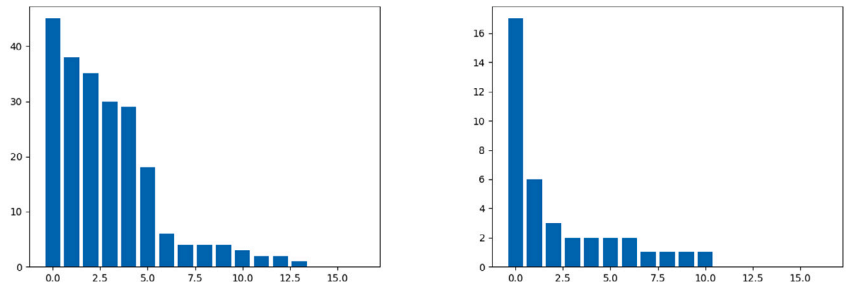
Fig. 3. Number of rules per conclusion element.

concern comment management, or related to login. We hereafter present a few examples of implications to illustrate how they carry meaning from the targeted domain (here online business):

- Between roles. $r1 \rightarrow r2$ states that when there is a triple $(s, r1, f)$ then there is a triple $(s, r2, f)$. E.g., for ActionVerb excerpt, when Game designers and Marketing managers have a feature in a system, Content creators also have it:
  $Game\ designer, Marketing\ manager \rightarrow Content\ creator$ (support = 3)
- Between features. $f1 \rightarrow f2$ states that when there is a triple $(s, r, f1)$, then there is a triple $(s, r, f2)$. E.g., for ActionVerbResource excerpt, when a role can browse a product list and update a user profile in a system, they can also search products:
  $browse\_productlist,\ update\_userprofile \rightarrow search\_product$ (support = 43)
- Between pairs (role,feature). $(r1, f1) \rightarrow (r2, f2)$ states that when there is a triple $(s, r1, f1)$, then there is a triple $(s, r2, f2)$. E.g., for ActionVerbResource excerpt, when Content creators can add a product and Final users can pay, then Final users can search products
  $(Content\ creator; add\_product), (Final\ user; pay\_shoppingcart) \rightarrow (Final\ user; search\_product)$ support = 31:



(a) Number of role-conditioned rules between features, each feature being in the conclusion of: ActionVerb (left-hand side), ActionVerbResource (right-hand side). Features have been ordered by decreasing number of rules.



(b) Number of feature-conditioned rules between roles, each role being in the conclusion of: ActionVerb (left-hand side), ActionVerbResource (right-hand side). Roles have been ordered by decreasing number of rules.

Fig. 4. Number of conditional rules per conclusion element.

- Conditional, with features as the conditions. $(A \to B)_C$ states that all systems that offer the features of C to the roles of A also offer them to the roles of B. E.g., for ActionVerbResource excerpt), if a system provides $add\_announcement$ to an $Intermediate\ seller$, it provides it to a $Marketing\ manager$:
  $(Intermediate\ seller \to Marketing\ manager)_{add\_announcement}$ (support = 3)
- Conditional, with roles as the conditions. $(A \to B)_C$ states that for all systems that offer the features of A to all roles in C, the features of B are also offered. E. g., for ActionVerb excerpt, if a system provides $browse, login, pay$ to a $Final\ user$, it also provides them $subscribe$:
  $(browse, login, pay \to subscribe)_{Final\ user}$ support $= 33$

Another way to analyze implications coming from the proper premise base consists in identifying binary implications and equivalences. The latter correspond to co-occurrences:

- Equivalence of support 2 in ActionVerbResource excerpt:
  $update\_advert \leftrightarrow delete\_advert$
- Equivalence of support 3 in ActionVerb excerpt:
  $(Game\ designer; add) \leftrightarrow (Final\ user; signup)$

The number of implications in which a feature is located in the conclusion informs to which extent the presence of this feature depends on the presence of others. For example, in the excerpt ActionVerbResource, after removing implications with support < 10, these three features still appear in the conclusions of a significant number of implications:

- $search\_product$ is in the conclusion of 45 implications, whose average support is 20 (in the whole PPIB it is the conclusion of 253 implications)
- $subscribe\_user\ account$ is in the conclusion of 23 implications, whose average support is 15 (in the whole PPIB it is the conclusion of 335 implications)
- $update\_user\ profile$ is in the conclusion of 24 implications, whose average support is 13.5 (in the whole PPIB it is in the conclusion of 332 implications)

All these examples illustrate how many patterns with significant support and valuable content can be extracted using TCA.

4.4.2. Threats to validity

Internal validity Internal validity relates to possible concerns and biases in data collection and reliability of tools. A first concern is quality and realism. The major part of the dataset comes from the front-end of real online commercial websites. The quality of manual building of the user-stories depends on the student understanding of the analyzed website, their skills in requirement modeling, and their point of view. The students found the exercise of identifying user roles and their handled features quite easy. But they had to extrapolate more for other roles, such as community manager, administrator, etc. They have been carefully supervised during the in-person classes, all along the extraction in order to ensure quality both during front-end user-stories extraction and back-end user-stories extraction or extrapolation. Such extractions are common in reverse engineering practice. A second concern is the variation on the results, on terms and depth level of the description. We addressed this threat by asking students to work in teams of 2-3 students, supervising this aspect of their work through general discussion with all the students in a class ($\sim$ 40 students), and by making a downstream manual alignment. The alignment itself is a subjective activity. A co-author of this paper, expert in the domain, has made an important manual work in revising the whole dataset, aligning the concepts and consolidating the quality of the result. Another co-author made an additional review. A third concern is the use of chatGPT. In the last months, the use of Large Language Models has become a very common practice for the software engineers experts in the industry. Our intent by allowing students to use it was to bring more realism to our methodology. To build their dataset, we noticed that students have yet a very minor use of chatGPT. A last concern is about the student skills and involvement. In software engineering research, many datasets are built on top of student work for two main reasons: (i) as they know that the research outcomes are most of the time soon available in software industry and they have a specific time dedicated to achieve this work, they constitute a motivated audience; (ii) as beginners, they will be the primary targets of the assistant tools and it is relevant to initiate the approach based on their skills, before adding more subtleties for experts.

Construct validity Construct validity relates to the relevance of the metrics we use to assess conclusions. This is quite easy to be convinced that measuring running time and implication number addresses scalability. Measuring the value of the extracted patterns is less easy. The support metric is commonly used in data mining, for implication and for association rules as well. The number of implications per conclusion is less frequent, we consider it here as an indicator of importance and generality of the conclusion, based on the intuition that the more a conclusion is in rules, the more it is shared.

Conclusion validity Conclusion validity assesses whether the conclusions derived from our results are reasonable. Obtaining a large number of rules and having a medium to high running time was an expected result, as the dataset has a significant size, with non neglectable variability and methods based on dyadic/triadic concept analysis are known to be combinatorial. The proper premise base is not a cardinality minimum set of rules, thus the high number of rules that we obtain makes sense. The qualities of this base rely on the minimal cardinality of the premises and on the unique element in conclusion, ensuring good readability. Many implications have a support of 0 or 1. This has been observed in other datasets in other domains. Implications with support 0 reveal things that are not observed, which are many. Implications with support 1 describe information which is specific to a single system, and we can expect they are many as well. The domains being close (leisure websites), and the vocabulary being carefully aligned, observing a significant number of commonalities is not surprising.

External validity External validity assesses to which extent results can be generalized to other situations. In our case study, we studied a specific category of software, i.e. commercial websites applied to leisure domains. Changing the type and the domain of software may have an impact on the running time, result size, and on what we learned. To address at best this threat, we considered four different domains and a significant number of websites in each domain. As it has been said, this case study is exploratory, and would deserve to be enlarged.

5. Related work

This section first presents work conducted in the domain of multi-dimensional SPLs, then approaches using FCA in this subdomain of Software Engineering. It concludes with a view on Triadic analysis and a review of practical applications of triadic implications to real data sets.

Considering several dimensions in SPLs In the SPL field, multiple dimensions have been studied in Multi Product Lines, n-dimensional product lines, and through the study of *variability in space and time* which is gaining attention. Multi Product Lines (MPLs) is a field of research that addresses the development of complex software ecosystems with a variety of platforms and involved organizations [22]. Several MPLs issues appear to be connected to ours. One of them is the presence of multiple stakeholders that separately or collaboratively may design or configure the product lines. A classical problem is then composing these designs afterwards. In our work, these stakeholders are not partners in the design and configuration of the product line as they are in the context of MPLs. They are rather collaborators that will use a derived product. Another close topic is the notion of n-dimensional product lines, which is developed in [23], as a complement to hierarchical structuring. They help structuring complex product lines. For example, mobile robot product lines can be structured along hardware dimension or behavior dimension. Constraints may arise inside a dimension or between several dimensions. Authors propose a structuring strategy, but they do not propose a solution to extract the variability constraints as we do in this paper. In our work, the dimensions are not all related to features, as one corresponds to the roles. Variability in space (i.e. through concurrent *variants*) and time (i.e. through successive *versions*) is currently attracting much attention in SPLs, as it is challenging to manage both in an integrated software process [24]. The approach of [25] aims to check consistency between models (e.g. feature model and source code) and to propagate the changes made. Hyper-Feature Models (HFMs) extend feature models with the various versions of each feature, where feature-aware constraints are completed by version-aware constraints [26]. These works do not address the question of extracting 3-dimensional logical constraints, which, to our knowledge, was up to now an unexplored research question in the SPL field.

FCA in software product lines and logical constraint extraction In the SPL field, FCA has been used for variable architecture extraction [27], feature location in families of similar software [28–30], or feature model composition [31], and to extract logical relationships. It is leveraged to classify feature usage, e.g. "always used" or "mutually exclusive" in [5]. The Duquenne-Guigues base of implications (DGBI) is also computed, to integrate additional constraints. In [6], attribute concept graphs serve as a basis to extract child-parent relationships corresponding to binary implications, mandatory/optional indications, and groups (or/xor). Authors use the PPIB to add the missing dependencies. We use this later base, but applied to 3-dimensional information, and we do not complete a feature tree. In [21], authors propose a labeled implication system derived from a feature model, which both captures relationships between features, and information about what is a valid and complete configuration. In all these proposals, only two dimensions are considered (configurations and features).

Extracting variability in the requirements with FCA has been more specifically made in [32,33]. In [32] the considered requirements are short sentences describing what the system can do, and the approach aims to synthesize a feature model. In [33], authors describe use-case diagram variants by their use-cases (features). In both works again, only two dimensions are taken into account (systems and requirements), and they do not use the actor (or role) dimension, as we do. In [9], authors consider variability in user-stories with the purpose of building feature models. They address the problem of the three dimensions in the user-stories, by reducing the descriptions into one two-dimensional table $System \times feature$ per role, and then one feature model per role. In the present paper, we conduct a prospective study considering the use of three dimensions as a whole in analyzing variability.

Triadic analysis and triadic rules in practice Triadic concept analysis has been introduced in [8] and generalized to the polyadic framework in [34]. Various aspects of triadic analysis have been explored. Fuzzy attributes are considered in [35]. Reduction is extended to TCA and applied in an oncological dataset in [36]. Basics, rules, factor analysis, and the fuzzy case are reported in [37]. Various types of triadic implications have been proposed in [16,17]. In [17], it is mentioned that they can be visualized in a concept lattice. A proposal for algorithms and conceptual navigation has been proposed in the FCA tools bundle collection [38]. Algorithms for extracting triadic associations, among others, are presented in [39] and tested on synthetic datasets. The evaluation shows the potential of triadic associations in terms of compactness compared to dyadic rules, and the feasibility in terms of computation time. Algorithms for computing precedence links between concepts, generators, and two kinds of implications in the triadic framework are proposed in [40]. Their computation time is evaluated on excerpts of the mushroom data-set,[4] the largest being composed of 1000 objects, 10 attributes and 4 conditions, with 125 implications (Biederman Cond. impl. and Biederman Attr. Cond. impl.) computed in 1.3 ms. Authors of [41] extend dyadic attribute exploration to the triadic case using the conditional attribute implications for the exploration. They aim to explore situations where multiple experts have different, and maybe contradictory views on a domain. A part of the Groceries database is analyzed in the triadic framework in [42]. The considered triples follow the pattern (customer, item, month). Authors compute implication and association rules among other triadic artifacts on 8121 transactions on 1000 customers, 30 items, during 24 months, with reasonable execution times on their platform, and they discuss a few of them [40]. TCA capacities for capturing a third, temporal, dimension, are used for longitudinal studies in [43,44]. In these two works, objects are individuals, attributes are different factors or behaviors and conditions are interview waves. Authors extract Biedermann Conditional Attribute Association Rules (BCAAR) and Biedermann Attributional Condition Association Rules (BACAR). Identifying rules related to long-lived profiles is more specifically addressed in [44]. The data-set is an excerpt from the population study ELSA-UK (English Longitudinal Study of Aging). The triadic analysis considers 50 long-lived individuals, 3 waves, and 13 factors. Using support and confidence above 50, authors obtained 52 BACAR rules and 1,968 BCAAR rules characterizing the long-lived profiles. In [44], the study aims to assist authorities to consolidate relevant policies to face the COVID-19 pandemic. The data-set collected in Nigeria is composed of 1800 persons (objects) interviewed about their mask usage, hand washing, medical treatment, and school activity (attributes) during three interview waves (conditions). When choosing a support greater than 50, and a confidence greater than 75, the authors obtain 12 BCAAR Rules (with 1 condition) and 11 BACAR Rules. These rules are compared to the reproduction rate at the same periods, providing a dashboard for observing the effect of policies and citizen behaviors. The temporal dimension is also considered in the prospective work of [45], where we proposed, on a fictive example, directions to analyze software configurations in space (along their features) and time (along their release) with TCA. The present paper builds proper premise implication bases of triadic implications, when [43,44] build triadic association rules. We do not consider time as a dimension as in [43–45] and we do not apply attribute exploration as in [41]. Our motivation is dealing with a real data-set of significant size composed of user-stories to assess the benefits of the triadic approach in software variability analysis.

[4] https://archive.ics.uci.edu/dataset/73/mushroom [Accessed 2024-02-22].

6. Conclusion

In this paper, we have presented a prospective study on extracting 3-dimensional variability from the user-stories sets of a family of similar software systems. Our approach considered the dimensions of systems, roles, and features. It used triadic concept analysis to get implication bases with particular properties and obtain variability relationships that cannot be acquired in a dyadic framework. We applied the approach to a dataset composed of user-stories sets manually extracted from 65 commercial websites. We could derive a significant set of patterns, in a feasible running time.

From this study, we can draw several perspectives. TCA is a particular instance of Polyadic Concept Analysis where any number of dimensions can be considered [34]. Polyadic Concept Analysis may help analyzing more dimensions on the user-stories: systems, roles, action verbs, but also resources, added information on the features and details on the purpose.

In parallel to studying the proper premise implication base, we considered using the Duquenne-Guigues base of implications (DGBI), another implication base which has a cardinality minimal set of implications. Due to the complexity of the DGBI computation, we faced expected running time issues that will need to be solved to explore this approach further.

By observing the implications computed using TCA, we envision several solutions to post-process and sort the implications for facilitating their analysis. The PPIB may contain redundant implications, i.e. implications that can be deduced from others, and a post-processing would consist in removing them. This is related to the fact that premises are minimal. Sorting the implications by support is a simple strategy but it does not convey meaning. We could use the content of the implications to cluster them by thematic and patterns, as it has been proposed in [46].

Another perspective of this work is establishing a complete methodology to derive, from the implications, either a software configuration, or a variability model (e.g. a feature model or a variable use case diagram) [13]. Deriving a software configuration may benefit from the fact that the proper premise base is direct, meaning that all attributes, that can be deduced from an initial set of attributes, can be deduced at once using the implication base. For example, if an application engineer chooses a set of pairs $(role, feature)$, they immediately know the pairs $(role, feature)$ that come along.

CRediT authorship contribution statement

Alexandre Bazin: Writing – original draft, Methodology, Conceptualization. **Thomas Georges:** Writing – original draft, Data curation. **Marianne Huchard:** Writing – original draft, Methodology, Conceptualization. **Pierre Martin:** Writing – original draft, Methodology, Data curation, Conceptualization. **Chouki Tibermacine:** Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marianne Huchard reports financial support was provided by French National Research Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by the ANR SmartFCA project,[5] Grant ANR-21-CE23-0023 of the French National Research Agency. We thank the students and the teachers involved in the building of the dataset, and the anonymous reviewers of previous versions of this paper, who helped to improve it.

References

[1] T. Kehrer, T. Thüm, A. Schultheiß, P.M. Bittner, Bridging the gap between clone-and-own and software product lines, in: 43rd IEEE/ACM Int. Conf. on Software Engineering (NIER) 2021, IEEE, 2021, pp. 21–25.
[2] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering - Foundations, Principles, and Techniques, Springer, 2005.
[3] ISO Central Secretary, Software and syst. engineering — reference model for product line engineering and management, Standard, Int. Organization for Standardization, Geneva, CH, Jan. 2015.
[4] A. Cockburn, Agile Software Development, Addison-Wesley, 2001.
[5] F. Loesch, E. Ploedereder, Restructuring variability in software product lines using concept analysis of product configurations, in: 11th Europ. Conf. on Soft. Maint. and Reeng., CSMR 2007, IEEE, 2007, pp. 159–170.
[6] U. Ryssel, J. Ploennigs, K. Kabitzsch, Extraction of feature models from formal contexts, in: 15th Int. Conf. on Software Product Lines, SPLC 2011 (vol. 2), ACM, 2011, p. 4.

[5] https://www.smartfca.org/.

[7] J. Carbonnel, M. Huchard, C. Nebut, Modelling equivalence classes of feature models with concept lattices to assist their extraction from product descriptions, J. Syst. Softw. 152 (2019) 1–23.
[8] F. Lehmann, R. Wille, A triadic approach to formal concept analysis, in: 3rd Int. Conf. on Concept. Struct., ICCS '95, in: LNCS, vol. 954, Springer, 1995, pp. 32–43.
[9] T. Georges, L. Rice, M. Huchard, M. König, C. Nebut, C. Tibermacine, Guiding feature models synthesis from user-stories: an exploratory approach, in: 17th Int. Working Conf. on Variability Modelling of Software-Intensive Syst., VaMoS 2023, ACM, 2023, pp. 65–70.
[10] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, D. Hughes, Using constraint programming to manage configurations in self-adaptive systems, Computer 45 (10) (2012) 56–63.
[11] K.C. Kang, FODA: twenty years of perspective on feature modeling, in: 4th Int. Ws. on Variability Modelling of Software-Intensive Systems, in: ICB-Research Report, vol. 37, Univ. Duisburg-Essen, 2010, p. 9.
[12] G. Bécan, M. Acher, B. Baudry, S.B. Nasr, Breathing ontological knowledge into feature model synthesis: an empirical study, Empir. Softw. Eng. 21 (4) (2016) 1794–1841.
[13] E.A.O. Junior, I.M. de Souza Gimenes, J.C. Maldonado, Systematic management of variability in UML-based software product lines, J. Univers. Comput. Sci. 16 (17) (2010) 2374–2393.
[14] B. Ganter, R. Wille, Formal Concept Analysis - Mathematical Foundations, Springer, 1999.
[15] U. Ryssel, F. Distel, D. Borchmann, Fast algorithms for implication bases and attribute exploration using proper premises, Ann. Math. Artif. Intell. 70 (1–2) (2014) 25–53.
[16] K. Biedermann, How triadic diagrams represent conceptual structures, in: 5th Int. Conf. on Concept. Struct., ICCS '97, in: LNCS, vol. 1257, Springer, 1997, pp. 304–317.
[17] B. Ganter, S.A. Obiedkov, Implications in triadic formal contexts, in: 12th Int. Conf. on Concept. Struct., ICCS 2004, in: LNCS, vol. 3127, Springer, 2004, pp. 186–195.
[18] K.M. Hill, In search of useful collection metadata: using openrefine to create accurate, complete, and clean title-level collection inf, Ser. Rev. 42 (3) (2016) 222–228.
[19] K. Murakami, T. Uno, Efficient algorithms for dualizing large-scale hypergraphs, Discrete Appl. Math. 170 (2014) 83–94.
[20] S.O. Kuznetsov, S.A. Obiedkov, Some decision and counting problems of the Duquenne-Guigues basis of implications, Discrete Appl. Math. 156 (11) (2008) 1994–2003.
[21] J. Carbonnel, K. Bertet, M. Huchard, C. Nebut, FCA for soft. prod. line representation: mixing configuration and feature relationships in a unique canonical representation, Discrete Appl. Math. 273 (2020) 43–64.
[22] G. Holl, P. Grünbacher, R. Rabiser, A systematic review and an expert survey on capabilities supporting multi product lines, Inf. Softw. Technol. 54 (8) (2012) 828–852.
[23] J.M. Thompson, M.P.E. Heimdahl, Structuring product family requirements for n-dimensional and hierarchical product lines, Requir. Eng. 8 (1) (2003) 42–54.
[24] S. Ananieva, S. Greiner, T. Kehrer, J. Krüger, T. Kühn, L. Linsbauer, S. Grüner, A. Koziolek, H. Lönn, S. Ramesh, R.H. Reussner, A conceptual model for unifying variability in space and time: rationale, validation, and illustrative app, Empir. Softw. Eng. 27 (5) (2022) 101.
[25] S. Ananieva, T. Kühn, R.H. Reussner, Preserving consistency of interrelated models during view-based evolution of variable systems, in: Proc. of the 21st ACM SIGPLAN Int. Conf. on Generative Programming: Concepts and Experiences (GPCE), ACM, 2022, pp. 148–163.
[26] C. Seidl, I. Schaefer, U. Aßmann, Integrated management of variability in space and time in software families, in: 18th Int. Software Product Line Conf. (SPLC), ACM, 2014, pp. 22–31.
[27] A. Shatnawi, A. Seriai, H.A. Sahraoui, Recovering software product line architecture of a family of object-oriented product variants, J. Syst. Softw. 131 (2017) 325–346.
[28] T. Eisenbarth, R. Koschke, D. Simon, Locating features in source code, IEEE Trans. Softw. Eng. 29 (3) (2003) 210–224.
[29] Y. Xue, Z. Xing, S. Jarzabek, Feature location in a collection of product variants, in: 19th Working Conf. on Reverse Engineering (WCRE) 2012, IEEE Computer Society, 2012, pp. 145–154.
[30] H.E. Salman, A. Seriai, C. Dony, Feature-to-code traceability in a collection of software variants: combining formal concept analysis and information retrieval, in: IEEE 14th Int. Conf. on Information Reuse & Integration (IRI) 2013, IEEE Computer Society, 2013, pp. 209–216.
[31] J. Carbonnel, M. Huchard, A. Miralles, C. Nebut, Feature model composition assisted by formal concept analysis, in: 12th Int. Conf. on Evaluation of Novel Appro. to Soft. Eng. (ENASE), SciTePress, 2017, pp. 27–37.
[32] M. Mefteh, N. Bouassida, H. Ben-Abdallah, Mining feature models from functional requirements, Comput. J. 59 (12) (2016) 1784–1804.
[33] R. Al-Msie'deen, A.H. Blasi, H.E. Salman, S.S. Alja'afreh, A. Abadleh, M.A. Alsuwaiket, A. Hammouri, A.J. Al-Nawaiseh, W. Tarawneh, S.A. Al-Showarah, Detecting commonality and variability in use-case diagram variants, CoRR, arXiv:2203.00312 [abs], 2022.
[34] G. Voutsadakis, Polyadic concept analysis, Order 19 (3) (2002) 295–304.
[35] R. Belohlávek, P. Osicka, Triadic concept lattices of data with graded attributes, Int. J. Gen. Syst. 41 (2) (2012) 93–108.
[36] S. Rudolph, C. Sacarea, D. Troanca, Reduction in triadic data sets, in: S.O. Kuznetsov, A. Napoli, S. Rudolph (Eds.), Proc. of the 4th Int. WS FCA4AI@IJCAI, vol. 1430, CEUR-WS.org, 2015, pp. 55–62.
[37] L. Wei, T. Qian, Q. Wan, J. Qi, A research summary about triadic concept analysis, Int. J. Mach. Learn. Cybern. 9 (4) (2018) 699–712.
[38] L.L. Kis, C. Sacarea, D. Troanca, FCA tools bundle - a tool that enables dyadic and triadic conceptual navigation, in: S.O. Kuznetsov, A. Napoli, S. Rudolph (Eds.), Proc. of the 5th Int. WS. FCA4AI@ECAI, vol. 1703, CEUR-WS.org, 2016, pp. 42–50.
[39] R. Missaoui, L. Kwuida, Mining triadic association rules from ternary relations, in: 9th Int. Conf. on Formal Concept Analysis, ICFCA 2011, in: LNCS, vol. 6628, Springer, 2011, pp. 204–218.
[40] R. Missaoui, P.H.B. Ruas, L. Kwuida, M.A.J. Song, Pattern discovery in triadic contexts, in: 25th Int. Conf. on Concept. Struct., ICCS 2020, in: LNCS, vol. 12277, Springer, 2020, pp. 117–131.
[41] M. Felde, G. Stumme, Triadic exploration and exploration with multiple experts, in: 16th Int. Conf. on Formal Concept Analysis, ICFCA 2021, in: LNCS, vol. 12733, Springer, 2021, pp. 175–191.
[42] P.H.B. Ruas, R. Missaoui, M.A.J. Song, L. Kwuida, Mining the groceries database using triadic concept analysis, in: RealDataFCA 2021@ICFCA 2021, in: CEUR Ws. Proc., vol. 3151, 2021, pp. 36–43.
[43] M.D.M. Noronha, C.N. Nobre, M.A.J. Song, L.E. Zárate, Interpreting the human longevity profile through triadic rules - a case study based on the ELSA-UK longitudinal study, in: 18th World Cong. on Med. and Health Inf. (MEDINFO 2021), in: Studies in Health Tech. and Inf., vol. 290, IOS Press, 2021, pp. 782–786.
[44] P. Lana, C. Nobre, L.E. Zárate, M.A.J. Song, Formal concept analysis applied to a longitudinal study of COVID-19, in: 24th Int. Conf. on Enterp. Inf. Syst., ICEIS 2022, vol. 1, SciTePress, 2022, pp. 148–154.
[45] A. Bazin, M. Huchard, P. Martin, Towards analyzing variability in space and time of products from a product line using triadic concept analysis, in: 27th Int. Syst. and Soft. Product Line Conf. - vol. B, SPLC, ACM, 2023, pp. 85–89.
[46] J. Saoud, A. Gutierrez, M. Huchard, P. Marnotte, P. Silvie, P. Martin, Explicit versus tacit knowledge in Duquenne-Guigues basis of implications: preliminary results, in: RealDataFCA 2021@ICFCA 2021, in: CEUR Ws. Proc., vol. 3151, 2021, pp. 20–27.