Variability-Driven User-Story Generation Using LLM and Triadic Concept Analysis

Alexandre Bazin[1], Alain Gutierrez[1], Marianne Huchard[1], Pierre Martin[2,3] and Yulin (Huaxi) Zhang[4]
[1]*LIRMM, Univ. Montpellier, CNRS, Montpellier, France*
[2]*CIRAD, UPR AIDA, F-34398 Montpellier, France*
[3]*AIDA, Univ. Montpellier, CIRAD, Montpellier, France*
[4]*EPROAD, Université de Picardie Jules Verne, Amiens, France*

Keywords: Software Product Line, Variability, User-Story, Requirements, Agile Product Backlog, LLM, Formal Concept Analysis, Triadic Concept Analysis.

Abstract: A widely used Agile practice for requirements is to produce a set of user stories (also called "agile product backlog"), which roughly includes a list of pairs (role, feature), where the role handles the feature for a certain purpose. In the context of Software Product Lines, the requirements for a family of similar systems is thus a family of user-story sets, one per system, leading to a 3-dimensional dataset composed of sets of triples (system, role, feature). In this paper, we combine Triadic Concept Analysis (TCA) and Large Language Model (LLM) prompting to suggest the user-story set required to develop a new system relying on the variability logic of an existing system family. This process consists in 1) computing 3-dimensional variability expressed as a set of TCA implications, 2) providing the designer with intelligible design options, 3) capturing the designer's selection of options, 4) proposing a first user-story set corresponding to this selection, 5) consolidating its validity according to the implications identified in step 1, while completing it if necessary, and 6) leveraging LLM to have a more comprehensive website. This process is evaluated with a dataset comprising the user-story sets of 67 similar-purpose websites.

1 INTRODUCTION

At the requirements stage, a widespread practice of the Agile paradigm is to provide a set of user-stories (also called "agile product backlog"), where a user-story is a brief sentence expressing the fact that a 'persona' (or role) wants to perform an 'action' (or have access to a feature) with a certain 'purpose' (Lucassen et al., 2016). In the context of Software Product Lines (SPL, (Pohl et al., 2005)), the requirements for a family of similar systems are therefore a family of similar user-story sets, one set per system. User-story sets are usually stored to support product line requirements documentation, guide the development, and are connected to the source code.

In this paper, we address the issue of *building the user-story set for a new system based on the variability logic of an existing system family, according to design options provided by the system designer.* We investigate a process that combines Triadic Concept Analysis (TCA) (Lehmann and Wille, 1995) and Large Language Model (LLM) prompting with the system designer input to suggest the user-story set for the new system. The design options of the new system are provided at an intermediate level of description (e.g. e-commerce), rather than at the feature level (e.g. pay), to alleviate the configuration work.

Our approach operates at the two stages of the traditional SPL framework (Pohl et al., 2005). It contributes to the *domain engineering* stage by building a variability model for requirements which is composed of (1) a set of triadic implications (Ganter and Obiedkov, 2004) and (2) an intelligible set of design options provided by LLM. At the *application engineering* stage, a selection of design options, made by the software designer, leads LLM to propose an initial user-story set. Then LLM uses the triadic implication set to consolidate the validity of the proposed user-story set, completing it if necessary to get a nearly valid configuration. Finally, we leverage LLM to propose user-stories related to the current user-story set, in order to have an even more comprehensive website.

Bazin, A., Gutierrez, A., Huchard, M., Martin, P. and Zhang, Y.
Variability-Driven User-Story Generation Using LLM and Triadic Concept Analysis.
DOI: 10.5220/0013360500003928
In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 618-625
ISBN: 978-989-758-742-9; ISSN: 2184-4895

The process is evaluated through a case study using a dataset of the literature (Bazin et al., 2024). Results are encouraging and indicate that the combination of the rigor of TCA and knowledge brought by LLM would be beneficial. This dataset is composed of user-story sets of 67 similar websites in several domains (mangas and derived products, martial art equipment, board games and video games).

Section 2 presents Triadic Concept Analysis (TCA) and the complexity for a software designer to leverage such outputs. Section 3 presents the approach. It outlines the process and presents the material and method adopted to address the case study. Section 4 presents and discusses the results. Related work is presented in Section 5, and we conclude in Section 6 with a summary and future work.

2 TRIADIC CONCEPT ANALYSIS

TCA in a Nutshell. Formal Concept Analysis (FCA) (Ganter and Wille, 1999) is a mathematical framework that aims at structuring information found in data in the form of binary relations. It starts with a binary relation where objects are described by attributes (see Table 1).

Table 1: A relation between systems as objects and features as attributes, inspired from (Bazin et al., 2024).

	search	*view comment*	*manage cart*
MyManga	×		×
MangaStore	×		
MangaHome		×	×

In binary relations, an implication is a pattern of the form $A \rightarrow B$ where A and B are attribute sets such that every object described by the attributes of A is also described by the attributes of B. For instance, the implication $\{view\ comment\} \rightarrow \{manage\ cart\}$ holds in Table 1 as all the systems offering the *view comment* feature (only *MangaHome*) also offer the *manage cart* feature.

The user-stories we consider are ternary relations between systems, roles and features (see Table 2). TCA (Lehmann and Wille, 1995) has been developed in order to exploit the more complex information they contain. In Table 2, a final user can search in all systems, and view comments only in *MangaHome*. A product manager can manage cart in *MyManga* and *MangaHome*, and view comments in *MangaHome*.

Implications in a triadic setting are more diversified than in their dyadic counterpart (Ganter and Obiedkov, 2004). Indeed, one can be interested in implications between features, between roles, or between the allocations of specific features to specific roles, *i.e.* pairs (feature,role) or symmetrically pairs

Table 2: A ternary relation between systems (*MyManga*, *MangaStore*, *MangaHome*), features (search *s*, view comment *vc*, manage cart *mc*) and roles (*FinalUser*, *Administrator*, *ProductManager*) (Bazin et al., 2024).

	s	*vc*	*mc*	*s*	*vc*	*mc*	*s*	*vc*	*mc*
MyManga	×			×		×			×
MangaStore	×			×					
MangaHome	×	×			×			×	×
	FinalUser			*Administrator*			*ProductManager*		

(role,feature). To obtain these latter rules, triadic data are brought back to a dyadic view: a binary relation is created by taking the Cartesian product of the required dimension as attributes and the Cartesian product of the other dimensions as objects. For instance, Table 3 depicts a binary relation between systems and the pairs (feature,role) they offer. The implication $\{(s,A)\} \rightarrow \{(s,FU)\}$ holds and means that all the systems that offer the *search* feature to administrators also offer it to final users. Two systems (*MyManga* and *MangaStore*) offer the implication premise (s,A). This number is called the *support* of the implication.

Table 3: A binary relation between systems and pairs composed of a feature (search *s*, view comment *vc*, manage comment *mc*) and a role (FinalUser *FU*, Administrator *A*, ProductManager *PM*).

	(*s*,*FU*)	(*vc*,*FU*)	(*mc*,*FU*)	(*s*,*A*)	(*vc*,*A*)	(*mc*,*A*)	(*s*,*PM*)	(*vc*,*PM*)	(*mc*,*PM*)
MyManga	×			×		×			
MangaStore	×			×					
MangaHome	×	×			×			×	×

In this paper, we use only implications between pairs (feature,role), and whose premise is a singleton, to prevent LLM from facing excessive computation challenges in this first study.

Handling Implications to Design a New System. Let's illustrate the limit of handling implications to design a new system using a small dataset taken from (Bazin et al., 2024), which introduces sets of user-stories from four manga-related websites. Two implications between pairs (role, feature) are shown below:

```
<4>  => (user;search)
<2> (communityManager;moderateComment)
    => (user;viewComment)
(...)
```

In this set, an implication is expressed as $< n > (r1; f1) => (r2; f2)$, where n is the support that informs on the number of websites which provide the premise $(r1; f1)$ of the implication. Such an implication thus means that in n websites, when role $r1$ can perform feature $f1$ (premise), then role $r2$ can perform feature $f2$ (conclusion).

User story sets and binary implications capture a large part of the variability logic of the Manga-related

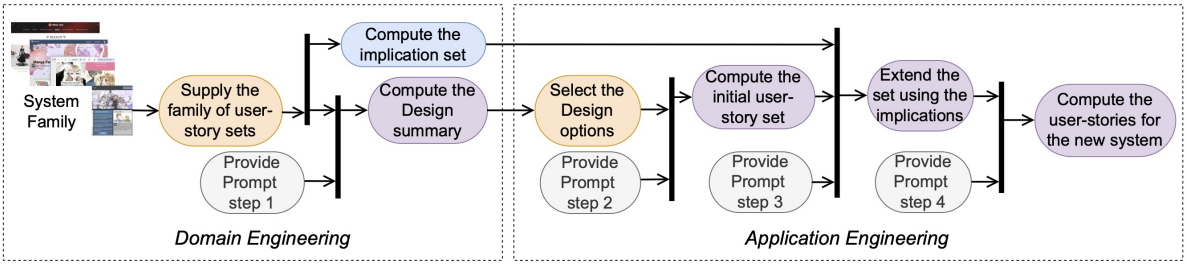

Figure 1: Overview of the process within the software product line framework. In this UML activity diagram, a color informs on the actuator: orange, violet, blue, and grey refer respectively to website designer, LLM, TCA, and prompt designer.

website family. They fix the vocabulary on role names (e.g. 'registeredUser' corresponds to 'premium user', 'subscriber', etc.), and feature names (e.g. 'CRUD-products' corresponds to 'manageProductsDB'). The implications indicate, for instance *all websites provide search to users* (1st implication, held by all 4 systems); *when a community manager can moderate comments, then users can view the comments* (2nd implication, held by 2 systems).

Implications being numerous, information is difficult to grasp by a software designer. Nor is it usable, as it does not give a synthetic report of the high-level options available (such as e-commerce or community management) and the logical dependencies to be respected when developing a new website. This is where LLM comes into play, with its ability to summarize and leverage knowledge to recommend features and dependencies in a more general setting.

3 APPROACH

Process Overview. Figure 1 outlines the process within the SPL framework (Pohl et al., 2005). The first activities take place at the *Domain engineering* stage, which focuses, in this work, on identifying commonalities and variability in the requirements of the systems provided as input. In a first step, the family of user-story sets is extracted from the system family storage, and then communicated with a prompt (Prompt step 1) to LLM. In return, the latter computes the main design options and provides a *design summary*. In parallel, the family of user-story sets is parsed with TCA, producing a set of implications that express logical dependencies between user-stories.

A second group of activities takes place in the *Application engineering* stage, which aims to produce a set of user-stories for the new system to be developed. As a second step of the process, the designers *select the design options* in the list proposed in the Design summary, then LLM provides the initial user-story set corresponding to this selection using Prompt step 2. Prompt step 3 asks LLM to extend the initial user-story set using the TCA implications. Finally, using Prompt step 4, LLM is requested to add or remove user-stories to get a more comprehensive system. All files of the case study are available online[1].

Tool and Dataset. The LLM adopted to conduct the case study is the general ChatGPT 4.0 model, to benefit from its latest enhancements. The purpose is to allow anyone to use or reproduce our results. Regarding the dataset, we used the file *ALL_System_Role_ActingVerb.csv* reported in (Bazin et al., 2024). This dataset gathers the sets of user-stories extracted in 2023 by students from 67 similar-purpose websites within the domains of mangas and derived products, sport equipment for martial arts, board games, and video games. The extraction has been supervised and the result has been reviewed and standardized. This dataset contains 1546 triples $(system, role, action\ verb)$, where 67 systems, 17 roles and 30 action verbs are involved, giving 91 user-stories. From this dataset, TCA computed 687 implications for the relation $system \times (role; feature)$ given in the companion repository[2].

Prompt Design. To write the prompt, we have complied with the recommendations of OpenAI[3] and literature (Schulhoff et al., 2024; Mondal et al., 2024; White et al., 2023). (i) A *role* (persona) is assigned to LLM in order to clarify its position in relation to the tasks it has to perform. (ii) A *context* is given as a key part of the prompt to guide LLM towards a relevant outcome. This context outlines the aim and the framework required by LLM to understand the data and the tasks. (iii) The chat is *decomposed* into several tasks. (iv) LLM is asked to *review* its answers. This often helps LLM correct mistakes and achieve a better result. (v) The *syntax* is explained and illustrated, notably when it is complex, such as implications, i.e. a

[1] https://doi.org/10.18167/DVN1/GNJMAV/
[2] https://doi.org/10.18167/DVN1/BWCC71
[3] https://platform.openai.com/docs/guides/prompt-engineering

mathematical logic formulation preceded by the implication support. Clearly identifying the syntax information on the prompt greatly improves the results. Finally, (vi) using *tags* allows to differentiate different parts within the prompt (e.g. `<Context>`, `<Syntax of data>`), and to refer data and results (e.g. `<User story data>`, `<Result step2>`).

The final prompt and the associated answers are presented below in a representative chat. As the aim of this section is to present the prompt, long answers are cut short and the removed parts are replaced by an ellipsis *(...)*. To facilitate data transmission to LLM, two files are uploaded in the chat, i.e. *UserStoriesInitial.csv* and *ImplicationsInitial.txt* that contain respectively the user-story set family for step 1 and the implication set computed by TCA for step 3.

Prompt step 1 asks LLM to provide a summary of high-level design options. This prompt is presented below.

> Step 1:
> <Role>: Your role is to assist software engineers in designing websites using the principles of software product lines. (...)
> <Context>: (...) you will be guided by a provided description of various similar websites in the targeted business domain (Manga websites) to identify the high-level design options. (...)
> <Syntax of data>: Each line in the raw data represents a user story of a website, expressed as a triple: "website, role, feature."
> <User story Data>: UserStoriesInitial.csv
> —
> <Task 1>: Based on <User story Data>, can you categorize these role-features into logical groups that represent distinct website feature modules (...)
> <Task 2>: Please review the groups to ensure accuracy, (...)

An answer excerpt to Prompt step 1, is as follows:

> (...) Proposed Feature Modules:
> *(...)*
> **Payment & Subscription Management**
> Roles: Finaluser, Subscriber, FinancialManager
> Features: pay, subscribe, unsubscribe, cancel
> *(...)*
> **Promotions & Marketing**
> Roles: MarketingManager, SocialMediaInfluencer
> Features: promote, advertise, campaign, evaluate
> *(...)*

Prompt step 2 is then completed by the selected Design options.

> Step 2:
> <Context>: In this step 2, you will identify the set of needed (role;feature) pairs to implement a set of chosen high-level design options (logical groups).
> —
> <Task 1>: Based on result of Task 2 of Step 1, can you list the (role;feature) pairs (i.e. the user-stories) that are needed to implement (...) [**Payment & Subscription Management and Promotions & Marketing**].
> <Task 2>: Review your result.
> <Formatted result step 2> Show the plain list of pairs (role;feature) from these groups (...)

In the answer, as presented below, LLM lists the relevant user-stories for the selected options.

> (Finaluser;pay)
> (Financialmanager;validate)
> (Intermediateseller;subscribe) *(...)*

In Prompt step 3, LLM is asked to consider the TCA implications in order to have a user-story set respecting the dataset logical dependencies:

> Step 3:
> <Context>: In this step, you will leverage logical dependencies (...) Using them, you will recommend an initial set of user stories (...)
> <Binary Implications>:
> In the general case, a binary implication follows the format "<n> (r1;f1) => (r2;f2) ", where (...)
> <Implications derived for Input Data>: ImplicationsInitial.txt
> —
> <Task 1>: The result of step 2 (<Formatted result step 2>) lists the pairs (role;feature) corresponding to the user stories associated to the high-level options chosen by the software engineer. Can you apply the <Implications derived for input data>, to recommend the set of user stories (pairs (r;f)) that describe this website? (...)
> <Task 2>: Review your result
> <Result 1 step 3> Show the plain list of applied implications (...)
> <Result 2 step 3> Show the final plain list of (role;feature) pairs (i.e. user stories). (...)

In its answer (see below), LLM lists the applied implications, and updates the user-story list.

> Result 1 step 3 (Applied Implications):
> (Finaluser;unsubscribe) => (Finaluser;add)
> (Finaluser;makeanoffer) => (Finaluser;add) *(...)*
> Result 2 step 3 (Final Set of (Role;Feature) Pairs):
> (Administrator;login) (Administrator;manage)
> (Contentcreator;add) *(...)*

Prompt step 4 asks LLM to enlarge the proposal:

> Step 4:
> <Context> In this step, we would like to refine the previous result (...)
> <Task> (...) would you add or remove any (role; feature) pairs in <Result 2 step 3>? (...)
> <Result step 4> Updated list of user-stories (...)

The final answer of LLM completes the website design, with explanations:

> (...) Here are the adjustments made:
> Additions:
> (Finaluser;download): Users might want the ability to download Manga chapters or entire volumes for offline reading, (...)
> *(...)*
> Removals: None; (..)
> Here's the final set of user stories for a comprehensive Manga website:
> Result Step 4:
> Administrator;login
> Administrator;manage
> *(...)*

Investigated Questions / Aim of the Evaluation. The case study aims to assess LLM ability: to produce a relevant set of user-stories for a new system in the SPL framework, and to combine its knowledge with logical dependencies extracted from existing systems using a logic-based method, i.e. TCA. We thus focused on these main research questions:

- (Q1) Is LLM able to properly summarize the shared or specific high-level design options of the existing system family?
- (Q2) Is LLM able to leverage logical dependencies to derive a nearly valid set of user-stories, starting from user-stories selected by the software engineer?
- (Q3) Is LLM able to extend the set of user-stories with proposals that make it more comprehensive, while avoiding straying too far from the initial requirements?

4 EVALUATION: FINDINGS AND DISCUSSION

In this section, we present and discuss the results obtained on twenty representative conversations with LLM, and then we discuss threats to validity.

Step 1. Design options summary. At step 1 of each of the 20 conversations, LLM answers a list of high-level options (the design options summary). The design summaries contain from four to eight options and have an average of 6.3 options. To assess the stability and content relevance of these 20 summaries, we developed a 2-prompt conversation launched five times. In the first prompt, we asked LLM to generate a report about the similarity of the 20 summaries. For evaluating this similarity, LLM has to identify common elements, based on identical names, synonyms, or terms with close semantics. In each of the five launched analyses, we observed that from six to eight options appear in more than half of summaries, revealing that most of the summaries share quite all their options. In the second prompt, we asked LLM to analyze more deeply the four most frequent options. The two most frequent ones are *User Management/Account Management* and *Content Management*. Then, these three following options appear in different orders: *Interaction/browsing* or (exclusive) *support/communication* and always *Subscription/Financial/Payment*. This reveals a stability in the summaries built by LLM. These elements support a positive answer to (Q1).

Step 2. List of user stories for the selected design options. The result of this step is rather straightforward to deliver for LLM, as it consists of enumerating the user-stories corresponding to the selection of one or more design options, that it created itself by grouping user-stories. Nevertheless, Table 4 shows that the number of user-stories grouped by LLM in a design option (at step 1) varies from one conversation to another, even if they were conducted the same day (e.g. Conversations Id 16, 17, and 18). This mitigates the positive answer to (Q1), as this means, that, even if nearly similar options names are presented to the system designer, these options may correspond to different user story groups.

Step 3. Application of the implications computed by TCA. Tables 4 and 5 report figures about the two results of step 3, i.e. applied implications and obtained user-stories respectively. Entrusting the task of applying the implications means that we are sufficiently confident in LLM ability to follow the application procedure described in the prompt and to enrich it. The set of binary implications we use has the property of being "direct" meaning that using the premises as input and applying the implications all at once provides all the user stories that can be inferred. This is an important property that eases LLM task. In order to assess our confidence, we developed a rule engine (*RuleEng*[4]) that applies TCA implications whose premise appears in step 2 result. The

[4]https://gite.lirmm.fr/gutierre/expeimplications

output of the rule engine is the set of deduced user-stories. We then compare the implications applied by *RuleEng* with those applied by LLM (Table 4), and the user-stories computed by *RuleEng* with those provided by LLM (Table 5).

Results show a difference in seven conversations between the implications applied by *RuleEng* and those effectively applied by LLM (in boldface in Table 4). Among these seven conversations, in conversations Id 11 and 14, LLM applies some new implications in addition to those applied by *RuleEng*. In the other five cases, there is a significant difference between implications applied by *RuleEng* and the ones applied by LLM, e.g conversation Id 4 for which only seven implications applied by *RuleEng* (on 95) were used by LLM among ten. Identifying the cause of such behavior raises questions (e.g. misunderstanding of the prompt or larger use of knowledge). The same evaluation was carried out for the computed user-stories (Table 5). This table shows 6 conversations where numbers differs, that also present a difference between implications, with a similar trend, i.e. when fewer implications are applied, fewer user-stories are computed, and reversely. Regarding the conversation that presents a difference in implications and not in the user-stories (conversation Id 11), we suppose that LLM did not apply some implications it declared to have applied. This result gives us a relative confidence in the way that LLM applies the implications and derives the user-stories, and contributes to answer partly positively to (Q2). We observe a significant number of conversations with low quality of implication application by LLM (about 1/3). A learned lesson is that, at this stage of LLM developement, after application of step 3 in real practice, it is recommended to compare the number of implications applied by both LLM and *RuleEng*. When the difference is significant, the designer can either discard the conversation, or try to redirect LLM.

Step 4. Upgrade of the user-stories using LLM. Table 6 reports the user-stories improvements made by LLM on the results from step 3. In four conversations (Id 8, 15, 16, and 17), user-stories were removed from step 3, meaning that LLM possibly considered some being semantic duplicates in the list. Of these four conversations, only one (Id 17) presents differences in both Tables 4 and 5. For all the conversations, we note that LLM adds user-stories. The increase ranges between 2% and 136%, and is 38% on average. A human reviewing confirmed their added-value, while remaining in the expected scope of the website domain, that fully justifies the use of LLM. For these conversations, we can conclude positively to (Q3).

Table 4: Implications (*Implicat.*) applied to obtain the set of user-stories in step 3 per conversation. Selected Design options are expressed by their acronym (e.g. T stands for Transaction). *US* stands for User-stories. Values in bold face highlight the differences between implications applied by LLM and the ones applied by RuleEng.

Conversation Id	Computation date	Selected Options in step 2	#initial US from step 2	#Implicat. applied by RuleEng	#Implicat. applied by LLM	#Implicat. applied by RuleEng and LLM
1	10/31	T	6	36	36	36
2	10/31	T/F	11	90	90	90
3	10/31	SP	8	69	69	69
4	10/31	SN/SI	13	**95**	**10**	**7**
5	10/31	PSM/PM	14	114	114	114
6	10/31	PSM/PM	22	104	104	104
7	11/02	FT	6	35	35	35
8	11/02	SPM	18	78	78	78
9	11/02	FT	7	**67**	**12**	**3**
10	11/02	MPF	8	**48**	**4**	**3**
11	11/02	FO	5	**42**	**53**	**42**
12	11/02	SPP	8	66	66	66
13	11/02	PSM	11	**67**	**8**	**6**
14	11/03	UIF	14	**46**	**112**	**46**
15	11/03	PT	8	64	64	64
16	11/03	TM	10	94	94	94
17	11/03	TM	10	**94**	**71**	**0**
18	11/03	TM	5	38	38	38
19	11/03	TF	9	65	65	65
20	11/03	SPM	6	63	63	63

Table 5: User-stories (*US*) computed at step 3 per conversation. Values in bold face highlight differences between US computed by LLM and those computed by RuleEng.

Conversation Id	Computation Date	Selected Options in step 2	#initial US from step 2	#US comput. by RuleEng	#US comput. by LLM	#US comput. By RuleEng and LLM
1	10/31	T	6	25	25	25
2	10/31	T/F	11	41	41	41
3	10/31	SP	8	36	36	36
4	10/31	SN/SI	13	**40**	**14**	**14**
5	10/31	PSM/PM	14	43	43	43
6	10/31	PSM/PM	22	41	41	41
7	11/02	FT	6	25	25	25
8	11/02	SPM	18	44	44	44
9	11/02	FT	7	**30**	**16**	**14**
10	11/02	MPF	8	**27**	**11**	**8**
11	11/02	FO	5	31	31	31
12	11/02	SPP	8	28	28	28
13	11/02	PSM	11	**33**	**11**	**11**
14	11/03	UIF	14	**27**	**50**	**27**
15	11/03	PT	8	28	28	28
16	11/03	TM	10	49	49	49
17	11/03	TM	10	**49**	**33**	**27**
18	11/03	TM	5	29	29	29
19	11/03	TF	9	28	28	28
20	11/03	SPM	6	31	31	31

Threats to Validity. Internal validity deals with datasets and tools quality. We refer the readers to the paper introducing the used dataset (Bazin et al., 2024) which exposes the concerns related to its building.

The uncontrolled element of this process is the LLM computation (e.g. summarizing), the fact that LLM parameters cannot be set in the current version we used, and knowledge it can bring. This corresponds to plausible current working condition for many software designers. In order to assess the ability of LLM to apply implications, we developed, apart from LLM, a rule engine named *RuleEng* to systematically apply implications and obtain the expected resulting user-stories. In addition, a systematic human review of LLM results ensured their coherency in relation with the task and input data (e.g. user-stories, implications). This systematic review also allowed to identify abnormal results, corresponding to a loss

Table 6: User-stories (*US*) per conversation obtained in step 3 and 4. Values in bold face highlight differences between US obtained by LLM in step 3 and those obtained in step 4.

Conversation Id	Computation Date	Selected Options in step 2	#initial US from step 2	#US listed in step 3	#US listed in step 4	#US listed both in step 3 and 4
1	10/31	T	6	25	37	25
2	10/31	T/F	11	41	55	41
3	10/31	SP	8	36	45	36
4	10/31	SN/SI	13	14	23	14
5	10/31	PSM/PM	14	43	52	43
6	10/31	PSM/PM	22	41	53	41
7	11/02	FT	6	25	33	25
8	11/02	SPM	18	**44**	**52**	**40**
9	11/02	FT	7	16	24	16
10	11/02	MPF	8	11	18	11
11	11/02	FO	5	31	38	31
12	11/02	SPP	8	28	35	28
13	11/02	PSM	11	11	17	11
14	11/03	UIF	14	50	57	50
15	11/03	PT	8	**28**	**35**	**24**
16	11/03	TM	10	**49**	**50**	**44**
17	11/03	TM	10	**33**	**78**	32
18	11/03	TM	5	29	38	29
19	11/03	TF	9	28	37	28
20	11/03	SPM	6	31	42	31

of quality of ChatGPT 4.o answers that has occurred during a short time, due to the change of its model. By nature of this tool, that shows randomness, we cannot have a perfect guarantee on the stability of the results and their repeatability.

We proposed various ways to assess the steps, i.e. a similarity study between the delivered design summaries for step 1, a comparison between the results of the rule engine and of LLM for step 3, and checking whether updates are of reasonable size and do not fall outside the domain scope for step 4. Designing more in-depth assessments remains a task for the future.

The case study deserves to be extended in several directions before generalizing (external validity), using a richer user stories description included in (Bazin et al., 2024), and considering other SPL domains. Nevertheless the study allows to expect that the approach is relevant on datasets of the same size and nature (commercial and community websites). We also could have considered other LLMs, but the objective was not to determine whether one model is better than another, but rather to demonstrate the feasibility of using an LLM.

5 RELATED WORK

LLMs provide many opportunities for achieving software engineering tasks, as it has been reported in a recent systematic literature review (Hou et al., 2024). Two works at the requirement stage are worth mentioning. An approach for synthesizing specifications of software configurations from natural language texts has been proposed in (Mandal et al., 2023). Here we do not rely on identifying specifications, as we dispose of user-stories, which are formatted expressions of specifications. LLM is used to evaluate the quality of user-stories in (Ronanki et al., 2024). In our present work, we do not evaluate the user-story sets and we consider they have a sufficient quality level to serve as a reference basis for building a new user-story set. A comparison of two approaches (rules versus LLM) to derive UML sequence diagrams from user stories is presented in (Jahan et al., 2024). Here, we do not aim to derive diagrammatic representations.

Domain models have been derived from user-stories using approaches including LLM interaction in (Arulmohan et al., 2023; Bragilovski et al., 2024). In (Bragilovski et al., 2024), examples of extracted domain concepts are personas, actions or entities. They used the reference dataset in (Dalpiaz, 2018), which contains user-story sets for single systems on different topics, and has been introduced in (Dalpiaz et al., 2019). In Step 1, we do not extract a domain model, rather we ask LLM to categorize the roles and features, thus to operate on this domain model to give a synthetic view of high-level design options. The dataset we use contains a family of user-story sets.

To our knowledge, there are few works that integrate SPL and LLM. One direction consists in applying Software Product Line Engineering (SPLE) principles to construct composite LLMs (Gomez-Vazquez and Cabot, 2024). In another direction, LLMs are used to achieve or assist with certain tasks of the SPLE life cycle, as we do in this paper. E.g. ChatGPT was used to synthesize SPL on the basis of a set of variants in (Acher and Martinez, 2023). In this latter paper, different types of system variants are considered: Java, UML, GraphML, state charts, and PNG. We follow this line of research with a few differences. Variability is identified using an exact method (i.e. TCA). When asking LLM to identify design options that group roles and features, the design options are a way to annotate the user-stories, which can be considered as a part of the product line to a certain extent. As suggested in the discussion in (Acher and Martinez, 2023), our proposal combines the use of LLM with a deterministic approach.

6 CONCLUSION

In this paper, we investigated the combination of LLM with a logical analysis method (TCA), applied to a user-story sets family in order to assist software engineers in the building of a new user-story set. The method uses (1) the knowledge extracted from the user-story sets family to frame the scope and guide towards valid configurations, and (2) knowledge of LLM to overcome the limitations inherent to the ex-

isting system family.

This work can be extended in several directions. First, TCA provides additional kinds of implications, not considered in this study, from which other types of logical dependencies (e.g. mutual exclusions) can be inferred. They can be used to fine-tune the software's final configuration. To address higher dimensions, like the purpose or the version, Polyadic Concept Analysis (Voutsadakis, 2002) can be used. Second, the process can be refined to better match designers' needs. For instance, LLM can propose various abstraction level options, or the implications provided by the rule engine can be used without requiring LLM to apply them. This may reduce the sensitivity of the configuration to the randomness of LLM.

ACKNOWLEDGEMENTS

This work was supported by the ANR SmartFCA project, Grant ANR-21-CE23-0023 of the French National Research Agency.

REFERENCES

Acher, M. and Martinez, J. (2023). Generative AI for reengineering variants into software product lines: An experience report. In *Proc. of the 27th ACM Int. Systems and Software Product Line Conf. - Volume B, SPLC 2023*, pages 57–66. ACM.

Arulmohan, S., Meurs, M., and Mosser, S. (2023). Extracting domain models from textual requirements in the era of large language models. In *5th Ws. on Artificial Intelligence and Model-driven Eng. @ ACM/IEEE MODELS 2023*, pages 580–587. IEEE.

Bazin, A., Georges, T., Huchard, M., Martin, P., and Tibermacine, C. (2024). Exploring the 3-dimensional variability of websites' user-stories using triadic concept analysis. *Int. J. Approx. Reason.*, 173:109248.

Bragilovski, M., van Can, A. T., Dalpiaz, F., and Sturm, A. (2024). Deriving domain models from user stories: Human vs. machines. In *32nd IEEE Int. Requirements Engineering Conf., RE 2024*, pages 31–42. IEEE.

Dalpiaz, F. (2018). Requirements data sets (user stories). Mendeley Data, V1, doi: 10.17632/7zbk8zsd8y.1.

Dalpiaz, F., Schalk, I. V. D., Brinkkemper, S., Aydemir, F. B., and Lucassen, G. (2019). Detecting terminological ambiguity in user stories: Tool and experimentation. *Inf. Softw. Technol.*, 110:3–16.

Ganter, B. and Obiedkov, S. A. (2004). Implications in triadic formal contexts. In *Conceptual Structures at Work: 12th ICCS 2004*, volume 3127 of *LNCS*, pages 186–195. Springer.

Ganter, B. and Wille, R. (1999). *Formal Concept Analysis - Mathematical Foundations*. Springer.

Gomez-Vazquez, M. and Cabot, J. (2024). Exploring the use of software product lines for the combination of machine learning models. In *Proc. of the 28th ACM Int. Systems and Software Product Line Conference*, SPLC '24, page 26–29. ACM.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., and Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.*, 33(8).

Jahan, M., Hassan, M. M., Golpayegani, R., Ranjbaran, G., Roy, C., Roy, B., and Schneider, K. (2024). Automated Derivation of UML Sequence Diagrams from User Stories: Unleashing the Power of Generative AI vs. a Rule-Based Approach. In *Proc. of the ACM/IEEE 27th Int. Conf. on Model Driven Engineering Languages and Systems*, MODELS '24, page 138–148. ACM.

Lehmann, F. and Wille, R. (1995). A triadic approach to formal concept analysis. In *3rd Int. Conf. on Conceptual Structures, ICCS '95*, volume 954 of *LNCS*, pages 32–43. Springer.

Lucassen, G., Dalpiaz, F., Van der Werf, J. M., and Brinkkemper, S. (2016). Improving agile requirements: the quality user story framework and tool. *Requirements Engineering*, 21.

Mandal, S., Chethan, A., Janfaza, V., Mahmud, S. M. F., Anderson, T. A., Turek, J., Tithi, J. J., and Muzahid, A. (2023). Large language models based automatic synthesis of software specifications. *CoRR*, abs/2304.09181.

Mondal, S., Bappon, S. D., and Roy, C. K. (2024). Enhancing user interaction in chatgpt: Characterizing and consolidating multiple prompts for issue resolution. In *21st IEEE/ACM Int. Conf. on Mining Software Repositories*, pages 222–226. ACM.

Pohl, K., Böckle, G., and van der Linden, F. (2005). *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer.

Ronanki, K., Cabrero-Daniel, B., and Berger, C. (2024). Chatgpt as a tool for user story quality evaluation: Trustworthy out of the box? In *Agile Processes in Software Engineering and Extreme Programming – Workshops*, pages 173–181, Cham. Springer Nature Switzerland.

Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., Gupta, A., Han, H., Schulhoff, S., Dulepet, P. S., Vidyadhara, S., Ki, D., Agrawal, S., Pham, C., Kroiz, G., Li, F., Tao, H., Srivastava, A., Costa, H. D., Gupta, S., Rogers, M. L., Goncearenco, I., Sarli, G., Galynker, I., Peskoff, D., Carpuat, M., White, J., Anadkat, S., Hoyle, A., and Resnik, P. (2024). The prompt report: A systematic survey of prompting techniques. https://arxiv.org/abs/2406.06608.

Voutsadakis, G. (2002). Polyadic concept analysis. *Order*, 19(3):295–304.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *CoRR*, abs/2302.11382.