Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPROACH TO ATTRIBUTED FEATURE MODELING FOR REQUIREMENTS ELICITATION IN SCRUM AGILE DEVELOPMENT

KARAM IGNAIM[1], SULTAN M. AL KHATIB[1], KHALID ALKHARABSHEH[1], JOÃO M. FERNANDES[2]

[1] Department of Software Engineering, Prince Abdullah bin Ghazi Faculty of Information and Communication Technology,
Al-Balqa Applied University (BAU), Jordan
[2] Departamento de Informática, Universidade do Minho, Portugal
E-mail: [1] karam.ignaim@bau.edu.jo, s.al-khatib@bau.edu.jo, khalidkh@bau.edu.jo, [2] jmf@di.uminho.pt

ABSTRACT

Requirements elicitation is a core activity of requirements engineering for the product to be developed. The knowledge that has been gained during requirements engineering about the product to be developed forms the basis for requirement elicitation. The agile approach is becoming known day by day as the most widely used innovative process in the domain of requirements engineering. Requirements elicitation in agile development faces several challenges. Requirements must be gathered sufficiently to reflect stakeholders' needs. Furthermore, because of the development process, requirements evolve, and they must be adequately treated to keep up with the changing demands of the market and the passage of time. Another challenge with agile implementation is handling non-functional requirements in software development. Addressing non-functional requirements is still a critical factor in the success of any product. Requirements prioritization is also one of the most challenging tasks, and it is uncommon for requirement engineers to be able to specify and document all the requirements at once. This paper presents an approach for requirements elicitation in scrum-based agile development. The approach operates with the feature modeling technique, which is originally used in the Software Product Line (SPL). One of the most important proposed extensions to Feature Models (FMs) is the introduction of feature attributes. Our method uses attributed FMs to consider both functional and non-functional requirements as well as requirement prioritization. For the evaluation purposes, we have demonstrated our approach through two case studies in different domains of software product development. The first case study is in the domain of education, and the second one is in the domain of health care. The results reveal that our approach fits the requirements elicitation process in scrum agile development.

Keywords: *Agile Methodology, Scrum, Attributed Feature Models, Requirements Elicitation, Requirement Engineering, Story Cards.*

1. INTRODUCTION

Requirement engineering is a process that aims to ensure the needs and goals of stakeholders for the product to be developed are well specified and articulated within a formal document [1]. This is to enable the reader, in particular the developer, to comprehend and specify the services that the product should include, their quality, as well as the constraints on the operation and development of the product. Requirement engineering involves five activities: elicitation, analysis, specification, representation, verification and validation, and management [1],[2]. Requirement elicitation is a fundamental and critical part of software development [3]. Previous research has also found that the requirements elicitation phase has a significant impact on requirement quality [4]. It involves discovering the user, system, and software requirements of a product from stakeholders [1]. The requirement can be a feature, functionality, behavior, or service that the user needs to have in the final product. The discovery of these requirements will lead to an understanding of what the software product does (i.e., functionality) and how well it performs against defined parameters (i.e., non-functional attributes). These aspects are worth looking at, as they will contribute to the success of any product development. Due to the increasing change in the customer's business needs and time to market, there is a need for adequate support by the development process for continuous communication with stakeholders and handling requirement change requests [5]. Unfortunately, traditional requirements engineering does not offer adequate support for these issues; thus, problems are exacerbated when new requirements are made due to changes in business needs and time to market [5]. For instance, the customers in the waterfall model provide their requirements in the initial phase before the development has started, which makes this model inflexible to deal with changes in the requirements.

The software industry is thus moving towards agile software development. This development methodology accommodates requirement changes, addresses changes during the development process, and speeds up the overall process. This is gradually established by this methodology based on its iterative nature. It allows for the capture and organization of an abstract version of the requirements, which then need to be prioritized according to their importance for the market and the product itself. Agile requirements engineering increasingly uses user stories, which is a concise notation format for expressing requirements [6],[7]. The most common format of a user story is: "As a (type of user), I want (goal), so that's some reason." [8], like "As a lecturer, I want to receive a notification when a student sends a message, so that I can respond to it." One of the best-known agile methods that incorporates the user story for requirement elicitation is called "scrum". In this method, software is delivered in chunks called "sprints". Accordingly, it enables engineers to adapt to evolving requirements as they change.

Actually, it is still difficult to determine which requirements should have a high priority and be included in early sprints [9]. Therefore, one of the most important and influential steps in the agile method of requirement elicitation is requirement prioritization. Prioritization of requirements is critical to the software team's understanding of the presence and significance of each requirement, its importance for use, and its urgency to market. Several studies have adopted search-based optimization algorithms in this regard. These algorithms provide useful tools not only for requirement prioritization but also for many software engineering problems, as in [10],[11],[12],[13]. However, different factors may influence requirement elicitation in software development, which requires the software team's attention. Another important factor that influences the quality of requirements elicitation is the inclusion of non-functional requirements. It plays an important role in agile methods for improving customer involvement in the development process. The addressing of non-functional requirements remains a key element of a successful software development project [14]. For that, it becomes necessary to investigate how agile practices handle non-functional requirements during requirements elicitation [14].

In spite of claims to the contrary, it is not "relatively easy" to specify non-functional requirements due to their inherent quantitative nature [15],[16]. In addition, because non-functional requirements are always tied to specific system functions, the quality of non-functional requirement approaches is frequently contingent on the quality of the requirement specification [17]. Moreover, the system quality and performance-specific issues compound the general requirements engineering difficulties: stakeholders frequently express their ideas vaguely, making it difficult to translate them into quantifiable and testable non-functional requirements; stakeholders are inaccessible; documentation is obsolete; and time and resources are limited in business practice [15]. Due to these difficulties and the shortcomings in eliciting non-functional requirements, agile software development methodologies are unable to effectively manage non-functional requirements [18]. Further challenges are reported for short iteration cycles, frequent deployments, and a lack of a comprehensive view of non-functional requirements by [18]. It has also been found by [19] that 26% of studies related to requirements reported the omission of quality practices during requirement specification. Consequently, non-functional requirements are often neglected in agile software development [19], [20].

Therefore, this paper provides a feature modeling approach (i.e., an attribute-based FM) for requirements elicitation using agile methodology (i.e., scrum methodology). The benefits of such an approach lie not only in capturing the functional requirements of the product to be developed but also in addressing the non-functional requirements and requirements prioritization during requirements elicitation of agile method. The approach makes use of the available tools that academics and business experts have recommended and used. The approach also facilitates the requirements elicitation process by enabling the prioritization of requirements and considering non-functional requirements.

The remainder of this paper is organized as follows. Section 2 provides the background used to develop the proposed approach while Section 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

discusses the problem statement. In section 4, the details of the proposed attributed feature modelling for requirement elicitation in Scrum development approach are provided. The supporting tools suggested are presented in Section 5. Moreover, the evaluation results are discussed in section 6. In Section 7, related works are presented. Finally, Section 8 concludes the paper.

2. BACKGROUND

In agile methodology, each user story represents a feature desired by the customer [1]. A feature represents a set of functionalities that are significant to a certain stakeholder(s) [21]. To present an artifact that can assist the stockholders during requirements elicitation and consider the issues of functional and non-functional requirements (features) and at the same time requirements prioritization, we use one of the major extensions' versions of the FMs technique, attributed FM [22]. Attributed feature modeling was adopted for its perfect match for the requirements elicitation in the agile methodology. In the context of variant families or so-called Software Product Lines, FMs have been frequently utilized to model commonality and variability [23] [24]. The primary criterion for using FMs is to derive a product configuration that meets all business and consumer needs. [25],[26]. As shown in Figure 1, feature diagram is the most commonly used visual notation for representing an FM [26]. The figure presents an example of the 'E-Shop FM' from the Feature IDE. A feature diagram represents features as nodes and relationships as edges. The root feature of the tree is called 'EShop'. There is always precisely one root feature present in every tree.

The FM supports the 'Catalogue', 'Payment', 'Security', and 'Search' features with optional support for 'Search'. The 'Security' is a common feature, and it is available either as 'High' or as 'Standard'. A feature can be represented as common /mandatory if its presence is required in the product. Contrarily, a feature that is said to be optional if its presence is optional in the product. Features can also be grouped in 'or' / 'or-groups' (like 'High' and 'Standard') or 'alternative'/ 'xor-groups' (like 'Bank transfer' and 'Credit card'). The dependency relationship between the features can be represented in the model either as "requires", in which selecting one feature implies selecting another, or as "excludes" in cases where two features mutually exclude each other. For example, 'Credit Card' implies 'High' security standard [18].

Extended FMs provide additional feature information via attributes. A feature's attribute is any measurable characteristic that can be used in crosstree relations [22]. For instance, Figure 2 shows a simplified version of the attributed FM for the E-Shop FM (presented in Figure 1 without attributes).

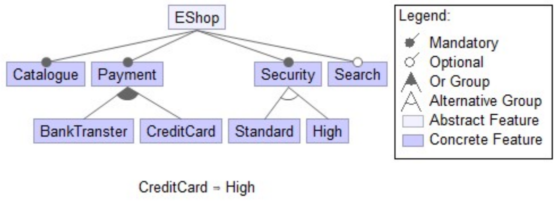
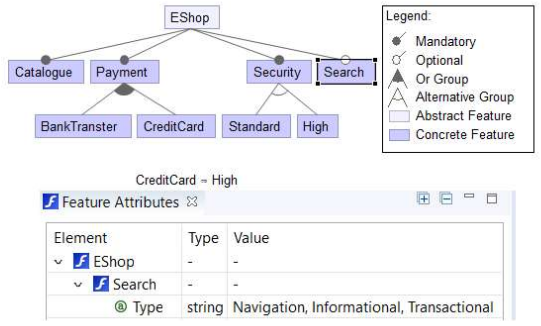


Figure 1 A Feature Model example [26].



Element	Type	Value
EShop	-	-
Search	-	-
Type	string	Navigation, Informational, Transactional

Figure 2 Attributed Feature Model example

This model is an extended FM due to the fact that some of the features contain attributes/parameters. The upper side of Figure 2 shows the feature diagram of the E-Shop FM, and the lower side of the same figure shows the feature attributes. The feature 'Search' has an attribute named Type with the domain: {Transactional, Navigational, and Informational}. So, for example, if we want to express the product that captures the user story card: "As a Web administrator, I want a navigation search type when a customer start to search, so the customer can get navigate through the products.", then the attribute/parameter features represent the product that consider all the common features, both payment methods (i.e., Credit card and Bank transfer), high security choice, and the user story card (that mentioned above) are: E-Shop-Product1-Navigational = {Catalogue, Payment, Bank transfer, Credit card, Security, High, Search}, where Search. Type= Navigational. If an urgent change request (in requirements) for Type attribute of the 'Search' feature is made, then this can be easily reflected, during requirements elicitation, using the attributed FM. For example, one of the products that derived from the E-Shop FM is E-Shop-Product1= {Catalogue, Payment, Bank transfer, Credit card, Security, High, and Search} and all the possible products that can be derived from this product and take attribute/parameter features into consideration are:

- ✓ E-Shop-Product1-Navigational = {Catalogue, Payment, Bank transfer, Credit card, Security, High, Search}, where Search.Type= Navigational.
- ✓ E-Shop-Product1-Informational = {Catalogue, Payment, Bank transfer, Credit card, Security, High, Search}, where Search.Type= Informational.
- ✓ E-Shop-Product1-Transactional = {Catalogue, Payment, Bank transfer, Credit card, Security, High, Search}, where Search.Type= Transactional.

3. PROBLEM STATEMENT AND OBJECTIVES

Scrum is a well-known agile development method that focuses on iterative product development [27],[28]. There are, of course, more agile methods, but scrum has become the most popular agile methodology among the agile family. This is almost because of its simplicity and the successful results achieved with scrum [28]. Figure 3 is a diagram of a scrum. Requirements elicitation in scrum is continuous, where a planning and daily scrum meetings are conducted. At the very beginning of each sprint, a planning meeting should be conducted. This improves the team's ability to respond in an agile manner to emerging challenges and enables the team members to create a sprint backlog. The sprint backlog, as shown in Figure 4, comprises a list of all the requirements that the development team should be working on during a particular sprint [9]. The requirements are prepared first in the form of user stories [8]. The PO makes the decision on requirement prioritization, where (s)he can offer comments on the requirements (features) that will be included in the upcoming sprint and receive regular updates on the development process. Normally, the daily scrum meeting should last no more than 15 minutes, which allows the PO to discuss any issues related to the requirements with the development team, decide to add a new requirement (feature) to the current sprint, and save time on reviewing the updates instead of asking for a change at the end.

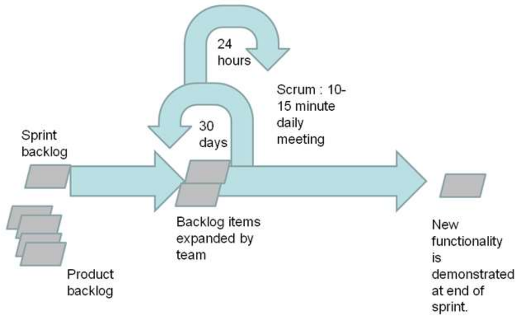

24 hours
Scrum : 10-15 minute daily meeting
30 days
Sprint backlog
Backlog items expanded by team
Product backlog
New functionality is demonstrated at end of sprint.

Figure 3 Scrum Methodology [1].

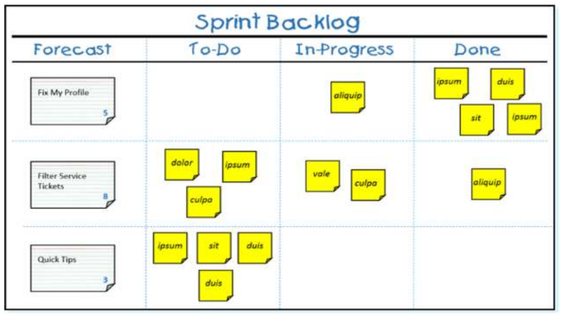


Figure 4 The sprint backlog.

At the end of each sprint, the team conducts a sprint review where they present the new assigned requirements (features) for the next sprint to the PO and stakeholders. In exchange, the PO and stakeholders provide their feedback for the next sprint. This feedback loop may lead to an alteration to the recently delivered functionality, but it is more likely to result in a review of the product backlog or the inclusion of new requirements. The product backlog is a complete list of the functionalities to be added to the product eventually, which are written as user stories and prioritized by the PO. Practically, the scenario that has been explained above suffers from two problems. As argued in the introduction, modern software development is going intensively towards adopting agile practices, so it becomes necessary for companies to handle non-functional requirements in agile software development, because they are not always obvious when dealing with functional requirements. It is also important to find a way to help stakeholders agree on the order of requirement prioritization. The source of the second problem comes from the fact that user stories are a widely used requirement notation in agile development. Unfortunately, the studies show that user stories are too often poorly written in practice and have inherent quality defects [8],[29]. Moreover, the daily meeting in scrum is too short and includes more than one party (i.e., stockholders), so an urgent artifact to bridge the gap between user stories and product backlog (i.e., to accelerate and facilitate requirement evolution and modification) is required. Triggered by the problems mentioned above, we have been motivated to propose an attributed-based feature modeling approach to tackle these problems. The approach is dedicated to requirements elicitation and evolution in scrum. Below is a summary of the objectives of our approach. Our approach supports the stakeholders who are involved in requirements elicitation: the PO, the Scrum Master (SM), requirement engineers, customers, and the development team. The objectives of the approach are to:

- Support the stakeholders for decision-making and information collection about the functional and non-functional requirements during the elicitation process. Please keep in mind that the approach does not cover additional requirement engineering tasks like modeling and analysis.
- Assist the PO in determining the eventual requirements that will be added to the product. In addition, it assists the PO in prioritizing the requirements so that the project team can consistently focus first on the most important, necessary, and relevant features.
- Allow requirement engineers to deal with requirements in the right way, handle changing requirements as they come up, and decide which requirements should be added in the early sprints.
- Help requirement engineers make good decisions about how to prioritize requirements in scrum processes.
- Help requirement engineers determine how many requirements they can handle at the time of planning meetings and daily scrum meetings. This helps them to better construct a sprint backlog.
- Help the PO keep track and make suggestions regarding the features that are being added during the development process.
- Help the PO save time by communicating with the development team before adding a new feature to a sprint, rather than requesting changes at the end.
- Assist the development team in conducting a quick sprint review at the end of each sprint, demonstrating feedback for the next sprint from both the PO and stakeholders.
- Assist requirement engineers in dealing with changing requirements and time constraints so that they can always focus on the most critical, necessary, and relevant features.
- Help the requirement engineers deal with non-functional requirements during the requirement elicitation.

4. ATTRIBUTED FEATURE MODELING BASED APPROACH

Due to the ideal connection between the attributed FMs and the proposed approach, we use this modeling technique to diagnose the problems to consider (1) user requirements (functional requirements, non-functional requirements, and constraints) in terms of features of the FMs and (2) and requirements prioritization in terms of features attribute/parameter. Please keep in mind that the researchers only employ feature modeling as a technique for requirement elicitation and not to represent the SPL. Figure 5 depicts an overview of the various activities involved in our approach's process (using a UML activity diagram), which works as follows:

- ✓ **Step 1.** The user stories were identified by stakeholders involved in the requirements elicitation of a particular product.
- ✓ **Step 2.** A requirement engineering expert starts from the user stories to *create attributed FM* for a specific product.
- ✓ **Step 3.** During the first meeting of each sprint, the development team, during the planning meeting, uses the attributed FM to *create the sprint backlog*.
- ✓ **Step 4.** The development team uses the attributed FM to *update the sprint backlog and product backlog*, during the daily meeting.
- ✓ **Step 5.** The PO makes use of the attributed FM at daily meetings in order to obtain constant updates regarding the development process and to supply feedback regarding the features that are being involved to the attributed FM.

This feedback loop leads to changes to recently delivered functionality, which in turn makes it more likely that new requirements will be reviewed or added to the product backlog. The development team can create and evolve the associated FM to reflect urgent changes in requirements whenever they wish. For example, the PO can use the attributed FM to decide which requirements should eventually be added to the product.

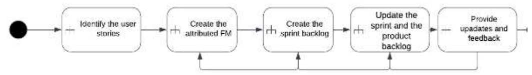

Identify the user stories
Create the attributed FM
Create the sprint backlog
Update the sprint and the product backlog
Provide upadates and feedback

Figure 1 The approach process.

4.1. Relating User stories and attributed FM

Figure 6 shows the user stories that appear as index cards. To create the attributed FM from this story card layout, the requirement engineering experts can translate the story card entries (i.e., *Title*, *Priority*, and *Description*) into an attributed FM as listed in Table 1. Figure 7 provides a concrete view of the attributed FM that represents Table 1. As shown in Table 1, according to our approach, each *Title* of the user story is mapped to a feature of the attributed FM and can be further characterized by mandatory or optional feature. The *Title* can be functional or non-functional requirements. Therefore, we cannot determine if a user story is mandatory or optional without knowing its context. The requirement engineer experts can decide this issue. This context is derived from the user story description and the user story's relationships to other user stories. For instance, if the functionality of a user story is always referenced by other user stories, then we can say that such a user story is mandatory. The *Priority* entry represents the attribute of the feature. The *Description* entry is also an attribute of the feature for nonfunctional requirements (i.e., response time). The requirement engineer experts can extract some useful information from description of the user story. This information can be used as attributes of the attributed FM, like cost, response time, CPU cycles, and memory size.

Table (1) The user story and the attributed Feature Model.

User story	Attributed FM
Title	Feature
Priority	attribute
Description (response time, CPU, …)	attribute

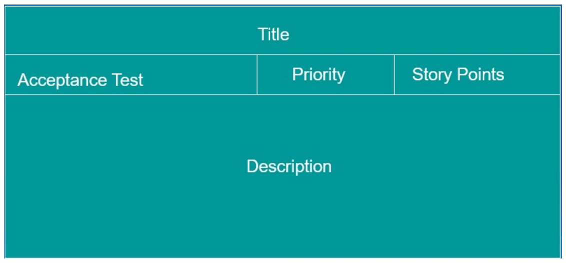


Figure 2 The story card layout.

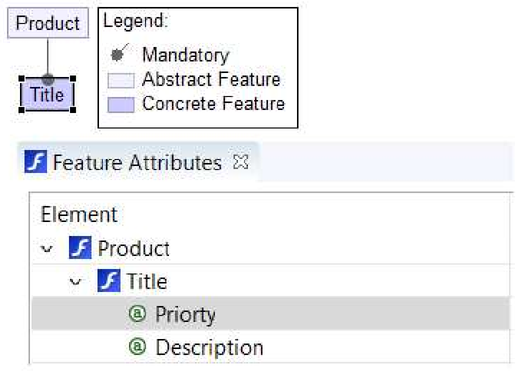
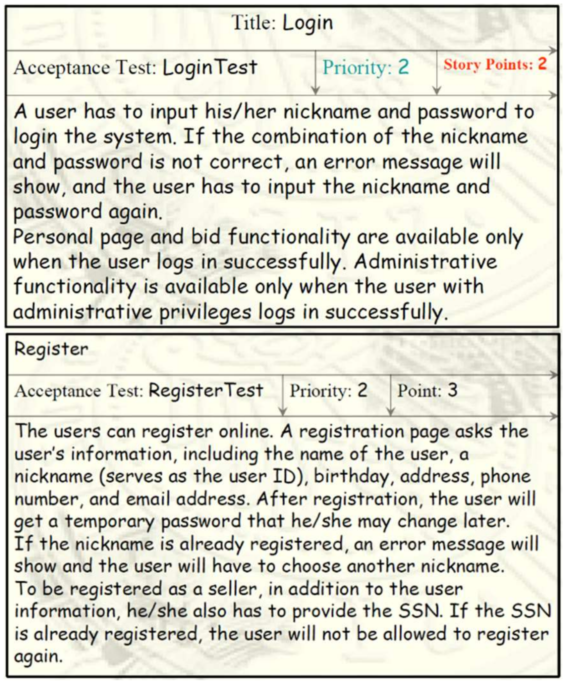


Figure 3 The attributed Feature Model that represents Table1.

4.2. An Illustrative Example

To demonstrate the approach, the attributed FM has been created from the story cards of the E-Shop web application. Figure 8 provides a partial view of user stories for this application. The attributed FM was developed using input gathered from brainstorming sessions with E-Shop web app stakeholders. The attributed FM for the E-Shop web app was developed by requirements engineer experts. To construct the FM, the engineer analyzed a huge number of story cards from webs in order to obtain the attributes used by online apps of this type. Figure 9 shows, for the sake of explanation, how the requirement engineer made the attributed FM of the E-Shop web app by using only the Login and Register story cards. Figure 10 depicts (a portion of) the attributed FM created by the engineers using story cards from the Shop web app. The Feature Diagram of the FM (the upper side of Figure 10) shows the functional features (e.g., 'Login', 'Register', and 'Update items') and the non-functional features (e.g., 'Transactions' feature with the attribute response time = 5 seconds, constraint on development language: 'Development' feature with the attribute Language = JSP) of the E-Shop App. It also shows the requirements prioritization (the lower side of Figure 10) as an attribute of features, for instance, the feature "Register" has a Priority attribute with a value of "1", whereas the feature "Update items" has a Priority attribute with a value of "3." This indicates that the "Register" feature has a higher development priority than the "Update items" feature.



Title: Login

Acceptance Test: LoginTest | Priority: 2 | Story Points: 2

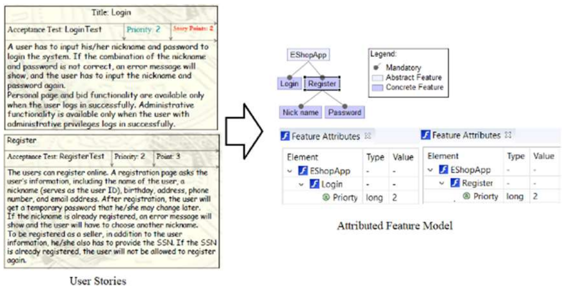
A user has to input his/her nickname and password to login the system. If the combination of the nickname and password is not correct, an error message will show, and the user has to input the nickname and password again.
Personal page and bid functionality are available only when the user logs in successfully. Administrative functionality is available only when the user with administrative privileges logs in successfully.

Register

Acceptance Test: RegisterTest | Priority: 2 | Point: 3

The users can register online. A registration page asks the user's information, including the name of the user, a nickname (serves as the user ID), birthday, address, phone number, and email address. After registration, the user will get a temporary password that he/she may change later.
If the nickname is already registered, an error message will show and the user will have to choose another nickname.
To be registered as a seller, in addition to the user information, he/she also has to provide the SSN. If the SSN is already registered, the user will not be allowed to register again.

Figure 4 Login and Register user story index cards.



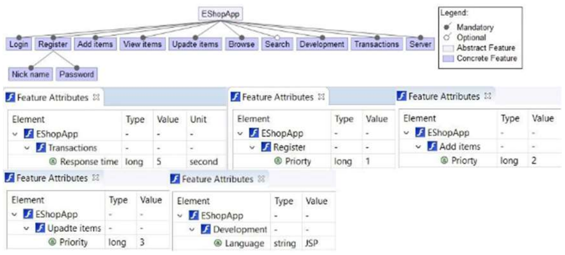
Feature Attributes

Element	Type	Value
EShopApp	-	-
Login	-	-
Priorty	long	2

Feature Attributes

Element	Type	Value
EShopApp	-	-
Register	-	-
Priorty	long	2

Figure 5 The attributed Feature Model of the E-Shop web application of Login and Registration story cards.



Feature Attributes

Element	Type	Value	Unit
EShopApp	-	-	-
Transactions	-	-	-
Response time	long	5	second

Feature Attributes

Element	Type	Value
EShopApp	-	-
Register	-	-
Priorty	long	1

Feature Attributes

Element	Type	Value
EShopApp	-	-
Add items	-	-
Priorty	long	2

Feature Attributes

Element	Type	Value
EShopApp	-	-
Upadte items	-	-
Priority	long	3

Feature Attributes

Element	Type	Value
EShopApp	-	-
Development	-	-
Language	string	JSP

Figure 6 The attributed Feature Model of the E-Shop web application.

5. TOOL SUPPORT

To support the methodology of our approach, we have embraced tools that are currently in use by academic researchers, business professionals, and other practitioners. This is for two reasons, the first one, generally, a fully-automated approach is technically not possible. The second is to permit requirements engineers and experts to intervene (manually) in our approach's process as needed. A user story is an informal description of a software feature that is written from the end user's perspective. To provide context to the development team, these stories should be written using non-technical language. Our approach gives stockholders the opportunity to work on existing tools that they have used to identify user story cards and sprint backlogs for a specific product. In addition, our approach uses a tool called Asana 1 for product and agile management (i.e., agile and scrum). Asana has both desktop and mobile apps (for Mac and Windows, but not for Linux). As shown in Figure 11, the Asana helps team members to manage the backlog to create stories and add them to the sprint. The Asana for desktop application is available and can be downloaded here 2. All actions and features available in the browser version of Asana are also available on desktop. To create the attributed FM, the requirement engineering experts can use the editor of the FeatureIDE, which is an open-source Eclipse-based platform for feature-oriented software development. Figure 12 shows the editor of the FeatureIDE. The experts in requirements engineering can use this editor to build the FMs. Figure 13 shows the window for feature attributes, where requirements engineers and experts can specify attributes for a feature of the FM. The experts can also use the constraints editor / view to view and manage the constraints of the FM, as shown in Figure 14.

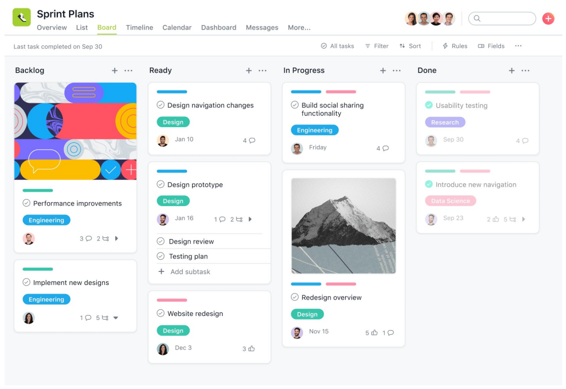


Figure 7 The Asana sprint plans.

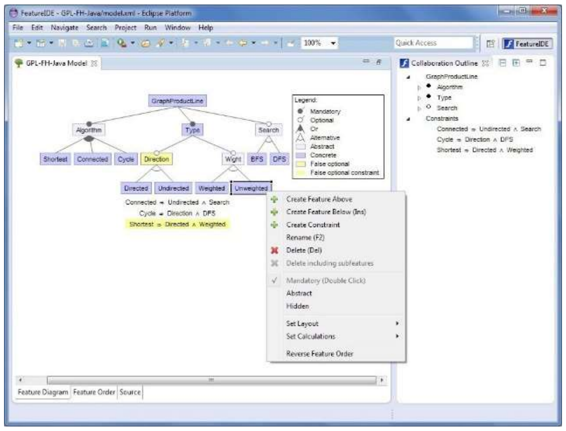


Figure 8 Feature model editor.

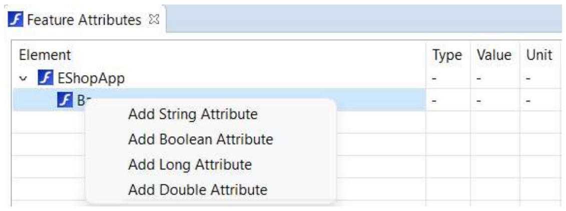


Figure 9 Feature attribute window.

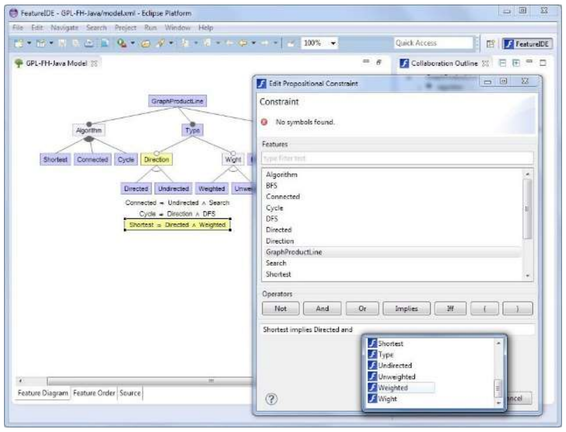


Figure 10 Constraint wizard.

6. EVALUATION

To evaluate our approach and estimate how the adopted tools fit our approach for the requirements elicitation, we decided to use case studies [30]. The use of case studies with real products/projects gives us the ability to assess the ability of the approach, as well as the adopted tools for requirements elicitation in scrum. In the evaluation of our approach, we were inspired by the work of Troyer [31]. We conducted two exploratory case studies to gain an initial understanding of the effectiveness and usability of the underlying approach and the tools adopted. One of the case studies is in the domain of education, and the second one is in the domain of health care.

6.1 CASE STUDY 1: EDUCATION DOMAIN

The first case study was non-formal. We plan to get introductory feedback on the usability of the suggested approach and the adopted tools in order to decide if it is worthwhile to continue with a formal case study. This case study was conducted with participants (i.e., students) of the Requirements Engineering and Formal Specifications course. As part of their project course, the students want to gather requirements for a Kids Math Game (KMG) app. They want to develop the app using scrum. The app is a game for the purpose of education through entertainment that helps kids to learn the basics of mathematical operations. The students were involved in an introductory session to explain our approach. The students involved in the evaluation have a background in the requirements elicitation of agile methodology. They gained this knowledge during the course. The majority of the KMG requirements that will be developed have already been discussed in one lecture of the course (one of the authors was a lecturer in the Requirements Engineering and Formal Specifications course). We concentrated on the usability and understandability of the tool during this evaluation session. In addition, its capabilities help students gather requirements successfully and effectively. During the evaluation of our approach, we (1) began by outlining the approach's objectives and giving a brief demonstration of the adopted tools for the participants (i.e., students). They (2) were asked to enter the user stories about the Kids Math Game app in the Asana tool during their behavior monitoring. We observed that the students were fully motivated and encouraged to work as a team. It took about 40 minutes to get started and enter the information. Following that, (3) we asked one of the students, who is an expert in requirement engineering, to create the attributed FM for the app. Finally, the participants (4) were asked to create a sprint backlog and product backlog based on the attributed FM. Finally, (5) we invited participants to an interview and solicited their feedback (on usability and understandability, as well as the relevance of the approach). This was done in a non-formal way during a course lecture. It was great hearing positive feedback from the students regarding the approach. The students were impressed by the proposed approach and the functionalities of the adopted tools, as well as by the completeness of the attributed FM (see Figures 15 and 16). They concluded that the approach and the adopted tools could be very useful in the requirements elicitation of requirements in scrum.

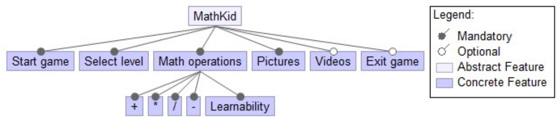
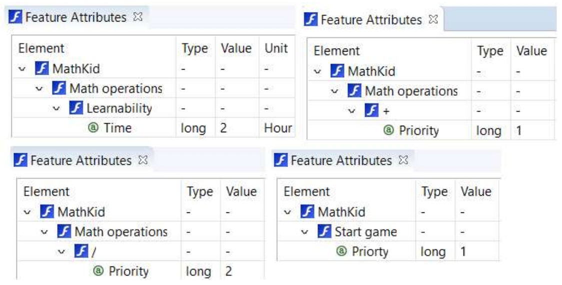


Figure 11 The attributed Feature Model of the Kids Math Game application.


Feature Attributes

Element	Type	Value	Unit
MathKid	-	-	-
Math operations	-	-	-
Learnability	-	-	-
Time	long	2	Hour

Feature Attributes

Element	Type	Value
MathKid	-	-
Math operations	-	-
+	-	-
Priority	long	1

Feature Attributes

Element	Type	Value
MathKid	-	-
Math operations	-	-
/	-	-
Priority	long	2

Feature Attributes

Element	Type	Value
MathKid	-	-
Start game	-	-
Priorty	long	1

Figure 12 Feature attributes for the Kids Math Game application.

6.2. CASE STUDY 2: HEALTH CARE DOMAIN

The second case study was performed with the participants (i.e., the development team) of a company that wanted to develop the COVID-19 Action Platform project. The participants have experience with developing products in scrum. We were looking for feedback on the learnability and usability of the underlying approach for this type of user because the participants were familiar with agile methodology and its related tools. The participants had no reason to be biased because they were simply concerned with business values. The objective of this case study is to evaluate our approach in the health care domain. Hence, we formulated the following hypotheses to measure the usability of our approach:

- ✓ H1: The feature-based approach and the adopted tools assist participants through requirements elicitation (i.e., functional and non-functional requirements) in scrum-agile development.
- ✓ H2: The feature-based approach and the adopted tools assist participants in prioritizing requirements in scrum-agile development.

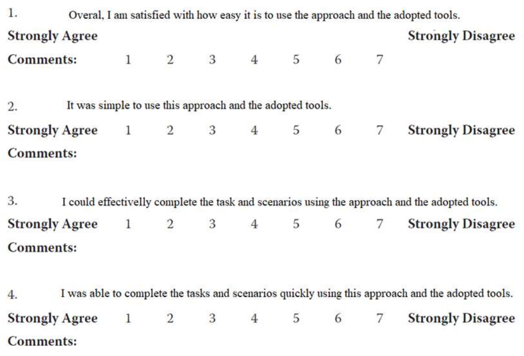
In this case study, we explained to the participants the attributed FM, and we fed the participants a brief explanation regarding the objective of the approach and how to use it for their own requirements elicitation. Additionally, we asked if they would be open to being interviewed and filling out a questionnaire afterward. After the team finished the requirements elicitation process, using our approach, we asked them to fill out a usability survey that contains learnability and usability-related questions to the participant pools (i.e., customers, PO, and team members) after having them experience the approach. A usability survey often includes questions involving the ease of use, ease of learning, simple preference, and other questions specific to the given system. The IBM Usability Questionnaire (Figure 17) was used as a questionnaire in our case studies for this purpose. To adapt the questionnaires to fit the evaluation of our approach and the associated tools, we rephrased the questions (see Figure 17) by replacing the word 'system' with the phrase 'approach and the adopted tools'. The IBM Usability Questionnaire has a scale that ranges from 1 (highest score, strongly agree) to 7 (lowest score, strongly disagree). The questionnaire's outcomes are shown in Table 2. Overall, the scores were all positive. For Q1, the participants gave 1 two times, 2 three times, and 3 one time, as well as 4. These scores give a good insight into the usability and understandability of the approach and the adopted tools for requirements elicitation. The participants return a close score for Q2, which provides a positive indicator regarding the approach's simplicity as well as its understandability. The lower scores (3 and 4 of Q1 as well as 4 two times 375 of Q2) belong to the PO and customer. These scores were due to some known difficulties with the attributed FM for non-computing users. According to Q3 of the qualitative evaluation, our approach can be used effectively to elicit the requirements of the COVID Action Platform project; the participants score 2 four times and 1 only once. The scores for Q4 of the questionnaire show that, on average (score 2), the participants agree that they were able to complete the project's requirements elicitation quickly using the approach and related tools. A final step in the evaluation is carried out at the end, where we conduct an interview to find out more about what the individuals who participate think (i.e., their experience with gathering requirements and familiarity with attributed FM).

Table (2) The scores of the participants from the IBM Usability Questionnaire.

Role	Score			
	Q1	Q2	Q3	Q4
PO	3	4	2	3
Customer	4	4	2	2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Member1	1	2	2	2
Member2	2	1	1	1
Member3	2	1	2	3
Member4	2	2	2	2
Member5	1	3	2	1


1. Overal, I am satisfied with how easy it is to use the approach and the adopted tools.
Strongly Agree Strongly Disagree
Comments: 1 2 3 4 5 6 7

2. It was simple to use this approach and the adopted tools.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree
Comments:

3. I could effectivelly complete the task and scenarios using the approach and the adopted tools.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree
Comments:

4. I was able to complete the tasks and scenarios quickly using this approach and the adopted tools.
Strongly Agree 1 2 3 4 5 6 7 Strongly Disagree
Comments:

Figure 13 Excerpts from the IBM Usability Questionnaire for computer systems [32].

Overall, the interview validated the questionnaire's results. The participants had no problems starting to use the approach, and the concepts, information, and terminology were clear. During the evaluation of the approach, the PO asked for an urgent change in requirements (i.e., add new features). We asked the team members to use our approach and the adopted tools to respond to this change. Afterwards, we asked them, during the interviews, about the role of attributed FM (see Figures 18 and 19) in response to an urgent change in requirements, and the PO and (vast majority) the team members confirmed that the approach supports requirements evolution successfully. Furthermore, they confirmed the approach assisted add new features that normally take longer to be added and prioritized without using the attributed FM. Moreover, the participants confirm that the approach can help in the daily meeting, and the attributed FM can help to bridge the gap between user stories and product backlog (i.e., to accelerate and facilitate requirement evolution and modification). This allows the team to always work on the most necessary, relevant, and valuable features first and address new requirements as business needs change. All these findings eventually led to acceptance of the hypotheses (I.e., H1 and H2), which proves that our approach and the adopted tools assist requirement engineers through requirement elicitation and prioritization in the scrum-agile method. Moreover, the interview results reflect high satisfaction from the participants, which helped us find a new hypothesis related to our approach (i.e., simplicity).

The results of the two case studies that were used to test the proposed approach show that it is useful and meets the needs of users in the target domains. The findings in this study confirm the findings in [31] and [33] regarding the positive influence of using feature modeling techniques in this context, and they can be effectively used in future studies to achieve more precise results. Despite widespread agreement on the role of feature modeling-based techniques and their ability to improve requirements elicitation, all studies agree that there are some limitations to the results' generalizability. These limitations concern the nature of the evaluated case studies. For this purpose, we plan to replicate the study using more case studies. Moreover, extending the approach to additional software development methodologies (e.g., the Rational Unified Process). A graphical software application that may be utilized by individuals and in meetings should be carefully considered.

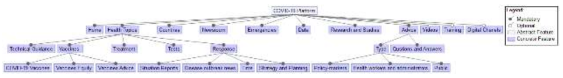
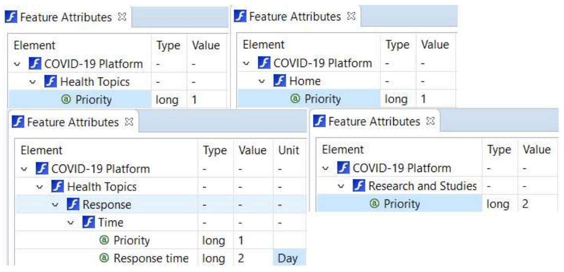


Figure 14 The attributed FM of the COVID-19 Action Platform.


Feature Attributes

Element	Type	Value
COVID-19 Platform	-	-
Health Topics	-	-
Priority	long	1

Feature Attributes

Element	Type	Value
COVID-19 Platform	-	-
Home	-	-
Priority	long	1

Feature Attributes

Element	Type	Value	Unit
COVID-19 Platform	-	-	-
Health Topics	-	-	-
Response	-	-	-
Time	-	-	-
Priority	long	1	
Response time	long	2	Day

Feature Attributes

Element	Type	Value
COVID-19 Platform	-	-
Research and Studies	-	-
Priority	long	2

Figure 15 The attributed FM of the COVID-19 Action Platform.

7. RELATED WORK

Several approaches have been proposed for requirement elicitation [8] [9] [31] [33] [34] [35] [36]. However, a few of them focused on using feature modeling techniques, such as [31] and [33]. In this section, the first part will highlight some studies that are relevant to requirement elicitation in general, and the studies that address feature modeling for requirement elicitation will be discussed in the second part.

In [8], the authors observed that user stories are a common way to write down requirements in agile development. Yet, user stories are not always written well and have problems with their quality. Because of this, the authors proposed the Quality User Story

(QUS) framework, which is a list of criteria that user story writers should try to meet. The authors also presented the Automatic Quality User Story Artisan (AQUSA) software tool, which was built on top of QUS. AQUSA finds quality problems and suggests ways to fix them by using natural language processing techniques.

In another work [9], the authors emphasized that there are various requirement prioritization techniques, strategies, and approaches, but the majority fail to account for classical metrics, such as ISO (9126, 25000) external metrics that affect software product quality. In this research, the authors considered ISO/IEC external metrics (25000, 9126) that affect process and product quality. As shown, these metrics (attributes) improve the quality of product requirements by considering all factors that affect prioritization, especially in ISO 25000. The study proposed a hybrid model based on factors, such as time to market, value, cost, market, risk, etc., combined with the influence of nonfunctional requirements over functional. It has all the beneficial elements of other techniques and a consistency check to assure that the proper requirements are chosen at the correct time in the scrum for the sprint under process.

Also, the authors of [34] published a systematic review study on the approaches and techniques used for automating requirement elicitation. The study covered more than 10 years of work in the domain and analyzed 68 papers. The results show that the automated requirements elicitation concentrates on humans as a source of data and exploits machine learning for data processing. They conclude that there is a need for more methods for requirements elicitation, in particular using big data, in order to develop high quality software.

Next, in [35], a framework for the gathering and concurrent collection of feedback and monitoring data has been introduced. The proposed framework supports continuous requirement elicitation in web applications and mobile applications. The strategy used the same ontology for monitoring data and managing feedback by using the same infrastructure. The findings show that the proposed method is effective in the industrial sector and assists businesses in understanding user needs. As a result, encourage and improve continuous requirements elicitation.

Recently, in [36], the authors worked to develop new methods of requirement elicitation based on the effects of culture, communication, basic competence, and stakeholder factors by integrating tools, processes, methods, and techniques to comprehensively solve the problems, and afterwards suggested an applicable framework. The authors conducted a long review, interviews analysis, observations, and groups from real-world projects specializing in software development in order to prove these effects.

On the other hand, the authors of [31] presented a methodology for eliciting domain-specific requirements. The authors believe that the stakeholders who should be included in the demand elicitation are increasingly from a variety of disciplines, backgrounds, and levels of software development expertise. This complicates the process of eliciting requirements. The authors have evaluated the methodology for two distinct application domains: serious games and e-shop web apps. Two exploratory case studies have resulted in a positive assessment of the technique and its related tool.

Also, a novel method called "PROPRE" has been proposed in [33]. The suggested technique combines current requirements engineering methods with a focus on practice. The method, which has been designed for industrial application, is simple to implement, and contradicts several commonly held beliefs. For instance, it does not require a comprehensive functional specification or a meeting of all stakeholders at the same time and location. The authors have effectively collected and specified the performance requirements of ABB's DDSS using requirements engineering. The authors were able to deliver a precise collection of performance criteria in a few weeks (two to three) through utilizing the method. The requirements are currently utilized to conduct routine performance tests in accordance with a test plan.

The main difference between this approach and previous works, in particular the works of [31] and [33], which are the most similar to our work, is that they focused on requirement elicitation. For example, in [31], they only addressed the domain requirements based on feature and variability modeling techniques, while in [33] the non-functional requirements. In [31], the proposed work was evaluated in two case studies from different domains serious games and web application while in

[33], only one case study was included from the automation domain. This work focused on the one hand, on functional and non-functional requirements elicitation in scrum-based agile development based on the feature modeling technique. On the other hand, improving the quality of scrum processes by prioritizing functional and non-functional requirements. Also, two case studies in the education and health care fields were used to evaluate the proposed approach.

8. CONCLUSIONS

In the agile software development process, changes in requirements must be addressed throughout the development process. Furthermore, requirements must be prioritized and managed with the utmost priority. In this paper, an approach has been proposed for requirements elicitation in the scrum-agile methodology. The proposed approach uses the feature modeling concept, where it models the issues to be considered during requirements elicitation by means of attributes/parameters of the FM. This improves the process as well as the product or project's quality. Our approach adopts tools that have been suggested and employed by scholars and industry professionals. Overall, the proposed approach and the adopted tools improve the ability of software engineers to always work on the most necessary, significant, and relevant features first or the new requirements that emerge as a result of changing business needs. The approach and the associated tools used were successfully evaluated through case studies in two different domains, namely education and healthcare. The findings reveal the proposed approach is usable and assists requirement engineers through requirements elicitation and prioritization in the scrum-agile method. Future work was presented, as well as the limitations of the approaches.

ACKNOWLEDGMENT

We would like to thank the Requirements Engineering and Formal Specifications course for the corporation in the case study.

REFERENCES

[1] P. A. Laplante, Requirements engineering for software and systems, Auerbach Publications, 2017.

[2] B. Nuseibeh, S. Easterbrook, Requirements engineering: a roadmap, in: Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 35–46.

[3] S. Tiwari, S. S. Rathore, A methodology for the selection of requirement elicitation techniques, arXiv preprint arXiv: 1709.08481 (2017).

[4] F. Anwar, R. Razali, Stakeholders selection model for software requirements elicitation, American Journal of Applied Sciences 13 (2016) 726–738.

[5] G. Waja, J. Shah, P. Nanavati, Agile software development, International Journal of Engineering Applied Sciences and Technology 5 (2021) 73–78.

[6] L. Cao, B. Ramesh, Agile requirements engineering practices: An empirical study, IEEE software 25 (2008) 60–67.

[7] X. Wang, L. Zhao, Y. Wang, J. Sun, The role of requirements engineering practices in agile development: an empirical study, in: Requirements engineering, Springer, 2014, pp. 195–209.

[8] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, Improving agile requirements: the quality user story framework and tool, Requirements engineering 21 (2016) 383–403.

[9] A. R. Asghar, S. N. Bhatti, A. Tabassum, Z. Sultan, R. Abbas, Role of requirements elicitation & prioritization to optimize quality in scrum agile development, International Journal of Advanced Computer Science and Applications 7 (2016).

10] Al Khatib, S. M. and Noppen, J. (2017). Benchmarking and comparison of software project human resource allocation optimization approaches. ACM SIGSOFT Software Engineering Notes, 41(6):1–6.

[11] Al Khatib, S. A comparative study of the relative performance and real-world suitability of optimization approaches for Human Resource Allocation. PhD thesis, University of East Anglia, (2018).

[12] Al Khatib, S. M. Optimization of path selection and code-coverage in regression testing using dragonfly algorithm. In 2021 International Conference on Information Technology (ICIT), pages
919–923. (2021). , IEEE.

[13] Al Khatib, S. M., Alkharabsheh, K., and Alawadi, S. Selection of human evaluators for design
smell detection using dragonfly optimization algorithm: An empirical study. Information and

Software Technology, 155:107120, (2023).

[14] S. Rahy, J. M. Bass, Managing non-functional requirements in agile software development, IET Software (2022) 60–72.

[27] K. Schwaber, Agile project management with Scrum, Microsoft press, 2004.

[28] H. Smits, G. Pshigoda, Implementing scrum in a distributed software development organization, in: Agile 2007 (AGILE 2007), IEEE, 2007, pp. 371–375.

[21] K. Ignaim, J. M. Fernandes, An industrial case study for adopting software product lines in automotive industry: An evolution-based approach for software product lines (evo-spl), in: Proceedings of the 23rd International Systems and Software Product Line Conference-Volume B, 2019, pp. 183– 190.

[22] A. S. Karataş, H. Oğuztüzün, Attribute-based variability in feature models, Requirements Engineering 21 (2016) 185–208.

[23] J. Bosch, Design and use of software architectures: adopting and evolving a product-line approach, Pearson Education, 2000.

[24] K. Ignaim, J. M. Fernandes, A. L. Ferreira, J. Seidel, A systematic reuse-based approach for customized cloned variants, in: 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC), IEEE, 2018, pp. 287–292.

[25] L. Machado, J. Pereira, L. Garcia, E. Figueiredo, Splconfig: Product configuration in software product line, in: Brazilian Congress on Software (CBSoft), Tools Session, Citeseer, 2014, pp. 1–8.

[26] K. Ignaim, Evospl: An evolutionary approach for adopting software product lines in the automotive industry (2021).

[30] J. Lazar, Jh feng e h. hochheiser, research methods in human-computer interaction, 2010.

[31] O. De Troyer, E. Janssens, A feature modeling approach for domain-specific requirement elicitation, in: 2014 IEEE 4th International Workshop on Requirements Patterns (RePa), IEEE, 2014, pp. 17–24.

[32] G. J. Kim, Human-computer interaction: fundamentals and practice, CRC press, 2015.

[33] R. Wohlrab, T. de Gooijer, A. Koziolek, S. Becker, Experience of pragmatically combining re methods for performance requirements in industry, in: 2014 IEEE 22nd International Requirements Engineering Conference (RE), IEEE, 2014, pp. 344–353.

[29]Alkharabsheh, K., Crespo, Y., Manso, E. et al. Software Design Smell Detection: a systematic mapping study. Software Quality Journal 27, 1069–1148 (2019).

[18] Brataas, Gunnar, Antonio Martini, Geir Kjetil Hanssen, and Georg Ræder. "Agile elicitation of scalability requirements for open systems: A case study." Journal of Systems and Software 182 (2021): 111064.

[15] Wohlrab, Rebekka, Thijmen de Gooijer, Anne Koziolek, and Steffen Becker. "Experience of pragmatically combining RE methods for performance requirements in industry." In 2014 IEEE 22nd International Requirements Engineering Conference (RE), pp. 344-353. IEEE, 2014.

[16] S. L. Pfleeger and J. M. Atlee, Software Engineering: Theory and Practice, 4th ed. Pearson, 2010.

[17] J. Dorr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, “Non-functional requirements in industry — three case studies adopting an experience-based NFR method,” in Proc. of the 13th IEEE Intl. Conf. on Requirements Engineering, 2005, pp. 373–382.

[20] Behutiye, W., Karhapaa, P., Lopez, L., Burgues, X., Martınez-Fernandez, S., Vollmer, A.M.,Rodrıguez, P., Franch, X., Oivo, M., 2020. Management of quality requirements in agile and rapid software development: A systematic mapping study. Inf. Softw. Technol. http://dx.doi.org/10.1016/j.infsof.2019. 106225.

[19] Ghanbari, H., Variainen, S., 2018. Omission of quality software development practices: A systematic literature review. Comput. Surv. 51.

[34] Lim, S., Henriksson, A. & Zdravkovic, J. Data-Driven Requirements Elicitation: A Systematic Literature Review. SN Computer Science 2, 16 (2021).

[35] Oriol, M., Stade, M., Fotrousi, F., Nadal, S., Varga, J., Seyff, N., & Schmidt, O. (2018, August). FAME: supporting continuous requirements elicitation by combining user feedback and monitoring. In 2018 ieee 26th international requirements engineering conference (re) (pp. 217-227). IEEE.

[36] Chaipunyathat, A., & Bhumpenpein, N. . Communication, culture, competency, and stakeholder that contribute to requirement elicitation effectiveness. International Journal of Electrical & Computer Engineering (2088-8708), 12(6), (2022).