Received May 2, 2020, accepted May 19, 2020, date of publication June 2, 2020, date of current version June 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2996951

Enabling the Reuse of Software Development Assets Through a Taxonomy for User Stories

EDNALDO DILORENZO, EMANUEL DANTAS, MIRKO PERKUSICH, FELIPE RAMOS, ALEXANDRE COSTA, DANYLLO ALBUQUERQUE, HYGGO ALMEIDA, AND ANGELO PERKUSICH, (Member, IEEE)

Electrical Engineering and Informatics Centre (CEEI), Intelligent Software Engineering (ISE) Group, Federal University of Campina Grande (UFCG), Campina Grande 58428-830, Brazil

Corresponding author: Ednaldo Dilorenzo (ednaldo.dilorenzo@virtus.ufcg.edu.br)

This work was partiatilly supported by CNPq and CAPES grants.

ABSTRACT *Context* - Agile Software Development (ASD) and Reuse-Driven Software Engineering (RDSE) are well-accepted strategies to improve the efficiency of software processes. A challenge to integrate both approaches is that ASD relies mostly on tacit knowledge, hampering the reuse of software development assets. An opportunity to enable RDSE for ASD is by improving the traceability between user stories (USs), the most used notation to register product requirements in ASD. Having enough link semantics between USs could enable defining similarity between them and, consequently, promote RDSE for ASD. However, this is an open challenge. *Objective* - To propose a taxonomy for adding link semantics between USs, focusing on easing the task of identifying similar ones. Such links, with support of traceability tools, enable the reuse of USs and their related assets. *Method*: We constructed a taxonomy for types of US focusing on Web Information Systems. The taxonomy is used to classify the US, given two facets: module and operation. Such information is used to infer the similarity between USs using link rules. We developed the taxonomy based on an empirical analysis of five product backlogs, containing a total of 118 USs. Afterward, we validated the taxonomy in terms of its potential to enable the reuse of US-related assets. First, we executed an offline validation by applying it to classify 530 USs from 26 already ended projects. Finally, we applied the taxonomy in a case study with two ongoing projects (59 USs). *Results*: The proposed taxonomy for USs is composed of two sub-facets, namely, module and operation, which have, respectively, three and 18 terms. In terms of coverage, for the offline study and case study, we classified 90.17% of the USs with the proposed taxonomy. For the case study, we classified all the USs analyzed. *Conclusion*: We concluded that it is possible to use our approach to compare USs and, consequently, retrieve their related assets. Our results regarding its practical utility have shown that users considered the taxonomy a useful approach to ease the process of assessing the similarity between user stories.

INDEX TERMS Agile software development, software reuse, user stories, information retrieval, technology acceptance model.

I. INTRODUCTION

Two well-accepted strategies that software companies can implement to preserve their competitive advantage, reducing their time to market, are Agile Software Development (ASD) and Reuse-Driven Software Engineering (RDSE) [1]. While ASD achieves this by having short validation cycles, incremental delivery and minimizing activities not directly related to executable code, RDSE reduces the effort necessary to produce artifacts by reusing existent knowledge (i.e., artifacts) such as source code, requirements, and test cases.

ASD is a change-driven approach to develop software. Its values and principles are stated in the Agile Manifesto [2] and applied through agile methods, such as Extreme Programming [3] and Scrum [4]. According to Hoda *et al.* [5], ASD has become ‘‘the mainstream development method of choice worldwide’’.

A characteristic of ASD is that it is mostly based on tacit knowledge [6] and frequent face-to-face communication, instead of explicit documentation. In ASD projects,

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Abid.

the main asset that stores product knowledge is the User Story (US) [7]–[9]. The US consists of a semi-structured notation to specify requirements, focusing on three aspects: the persona user, the functionality that the given user wants the system to provide, and the reasoning behind the user's desire. There are many templates available, but, according to Lucassen *et al.* [10], 70% of practitioners use the template proposed by Cohn [11]: "*As a <user>, I want <goal>, so that <reason>*".

The main goal of the US is to facilitate product-related discussions between the stakeholders. As a consequence, USs might not contain enough information to enable people not involved in the project to understand the project's scope in a low level of granularity (e.g., technologies, algorithms or design used, types of user interface developed). Thus, the US and its related assets such as the *task cards* and *Product Backlog* are information intended to be used locally by the project team(s), and not globally by the organization. As a result, from an RDSE perspective, a down point of ASD is the lack of enough explicit product knowledge, which hinders the traceability of software development assets, thus making information reuse a challenge.

Having explicit product knowledge with enough information enables the indexing of reusable assets such as source code, components, test cases, and specifications. For instance, tools such as Gitlab,[1] which integrates features of project management, version control, continuous integration, and continuous delivery, enables to easily establish traceability links between US and tasks, commits, builds and time estimations (and tracking). Systematic reuse can bring significant benefits, such as the increase in software productivity and software product quality [12]. By reusing requirements, there is the potential to have a baseline to address some interesting questions such as: "Who is the most efficient developer or team to implement this feature using Django?", "How has this requirement been tested in the past? Was it efficient? How can we do better?", "Which non-functional requirements are usually considered in the context of these requirements?", "Who should review the source code related to this requirement?", "Who should fix this defect?" and "Which components or APIs could we use to reduce time-to-market delivering this feature?".

Despite the potential of reusing requirements, especially, functional requirements (i.e., description of what the software should do [13]), their use is deficient in the industry. The exception is for companies that work on products on the same domain area, which is the case for software product lines [14] and Software Requirements Patterns [15]. Given their loose coupling to the domain, non-functional requirements and non-technical requirements have higher reuse frequency than functional requirements [16]. Despite this, if we analyze companies that work on products of the same type [17], for instance, Web Information Systems (WIS), and not necessarily from the same family, we can observe that they repeatedly work on very similar tasks such as "creating a login page" or "implementing CRUD operations". This observation raises the following research question: *is it possible to enable the reuse of functional requirements-related assets for systems of the same type in the context of ASD?*

A potential solution is to use requirements catalog [18]. A requirements catalog contains a set of related requirements, which might belong to the same domain or profile. Each requirement is uniquely identified and classified according to a set of attributes such as priority, criticality, risk, source, and type. From the attributes presented in Pacheco *et al.* [18], which are based on the IEEE Std. 1233, the most related to our interest is *type*. Given this attribute, each requirement might be categorized into one or more of the following types: (i) data update requirements, (ii) information requirements, (iii) query and reports requirements, and (iv) requirements for interaction with existing systems. From the viewpoint of our research question, the classification of a functional requirement as presented in Pacheco *et al.* [18] has as the main following limitations the high cost associated with constructing the catalog, considering the number of attributes necessary to be defined for each requirement and the tool support.

Another alternative is to create link semantics and make explicit links between artifacts. Such an approach improves traceability and enables assessing the similarity between requirements [19], [20].

Espinoza and Garbajosa [19] present a tool that supports manually adding link semantics for agile artifacts and trace automation. Elamin and Osman [20] present a traceability methodology, focused in ASD, that, based on link semantics, generates traceability links identifying reusable requirements and related artifacts. Such a solution is based on the agile Traceability Information Model (TIM), commonly supported by agile management tools (e.g., Rally[2]).

Both studies, Espinoza and Garbajosa [19] and Elamin and Osman [20], rely on a semi-automatic approach, in which link semantics are manually added by the software engineer and reuse is automated with the support of traceability tools. Further, they do not provide models, instructions, or guidelines on how to add the link semantics. Currently, there is limited work on automatically creating link semantics between requirements, and such studies [21], [22] assume the availability of extensive documentation, which might not be a fit for ASD, and focus on distinguishing between requirements and non-requirements [23], functional and non-functional requirements [24], and identifying defects (e.g., consistency or completeness).

To address the need for an innovative solution to enable the reuse of functional features in the context of ASD, we propose a taxonomy to add link semantics between USs. Such a link enables identifying similar USs, and with the support of traceability tools, automate the reuse of agile artifacts. The proposed taxonomy is tailored to a system type (e.g., WIS)

[1]https://about.gitlab.com/

[2]http://www.rallydev.com

and such a solution a well-established approach to support RDSE [25], [26].

To construct the taxonomy, tailored to WIS, we used a mixed-methods empirical approach based on [27]. First, we analyzed the product backlogs from five projects composed of 118 USs and 336 tasks from a real software company to identify patterns between the USs and tasks. As a result, we defined a two levels taxonomy to represent the USs.

We validated the proposed taxonomy in terms of potential to enable the reuse of US-related artifacts. For this purpose, we, first, validated the taxonomy using data from 26 already ended projects (530 USs). Finally, we validated the taxonomy through a case study with two ongoing projects (35 USs), including collecting data from nine software engineers that used the taxonomy through the Technology Acceptance Model (TAM) [28].

This paper is structured as follows. Section II formally states the problem address by this study. Section III describes the methodology executed to construct the taxonomy. Section IV presents the proposed taxonomy. Section 4 presents the results of the validation study. Section VI discusses the study's research questions and implications for academia and industry. Section VII discusses the study‘s threats to validity. Finally, Section VIII presents our final remarks and future work.

II. PROBLEM STATEMENT

Let US be a set of user stories $US = \{u_1, u_2, \ldots, u_i\}$, where $i = |US|$, and C a set of classification terms $C = \{c_1, c_2, \ldots, c_j\}$, where $j = |C|$. We define that a mapping of a user story u_k to C is given by $M : US \rightarrow C$.

In what follows, we define the properties of a mapping M :.

- **Coverage** (M_c): is given by the number of different user stories elements contained in M divided by $|US|$.
- **Granularity** (M_g): is given by the number of different classification elements contained in M.
- **Heterogeneity** (M_h): is given by the number of elements for each possible element of C.
- **Complexity** (M_x): is given by the Granularity divided by $|US|$.

To clarify the properties of given mapping M, consider the following scenario:

1) $US = \{u_1, u_2, u_3, u_4, u_5\}$,
2) $C = \{c_1, c_2, c_3\}$, and
3) $M = \{(u_1, c_1), (u_1, c_2), (u_2, c_1), (u_3, c_2), (u_4, c_3)\}$.

For M_c, notice that the tuples contained in M contains u_1, u_2, u_3 and u_4, but not u_5. Therefore, M_c is calculated in Equation 1.

$$M_c = \frac{4}{5} * 100 = 80\%. \tag{1}$$

For M_g, notice that the tuples contained in M contains all the elements of C (i.e., c_1, c_2, and c_3). Therefore, M_g is 3.

For M_x, notice that the tuples contained in M contains all the elements of C (i.e., c_1, c_2, and c_3). Therefore, M_x is calculated in Equation 2.

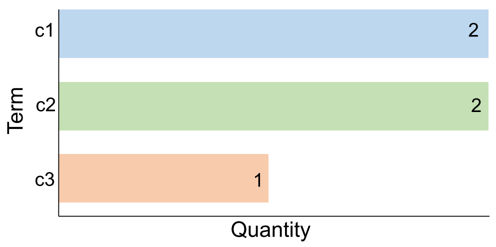


FIGURE 1. Distribution of the heterogeneity of M (M_h).

$$M_x = \frac{3}{5} * 100 = 60\%. \tag{2}$$

Finally, for M_h, notice that two tuples of M contain c_1 and c_2, and only one contains c_3. Figure 1 shows that distribution of M_h.

Let $S(u_i, u_j)$ be a function that calculates the similarity between user stories u_i and u_j and s be a set of similarities calculated for all user stories S $= \{s(a, b) \mid a \in US, b \in US$, and $a \neq b)\}$.

Let $T(US_i, SDLA)$ be a function that defines the relationship between the user story u_i and the set of software develpement lifecycle artifacts $SDLA$.

Let Q be the software developement lifecycle artifacts, where $Q \subset SDLA$, that can be reused for a user story u_i given a function $R = (T, S)$. R receives as input a u_i, and, given S, it identifies user stories $US_{similar_i}$ similar to u_i. Given T, it defines Q as the union of the related software development artifacts $SDLA_k$ for each $u_k \in US$.

This work mostly focuses on defining C. In other words, we focus on giving semantic links between user stories. More specifically, we focus in the domain of user stories for the Web Information Systems domain. The process for operationalizing M presented herein is a manual (see Sections IV and V). Further, for S we assume a simple rule presented in Algorithm 1.

Algorithm 1 Similarity Between User Stories u_i and u_j

```
1: procedure SIMILARITY($u_i$,$u_j$,$M$)
2:     if $u_i$ and $u_j$ are mapped to at least one common $c_k$ then
3:         return TRUE                    ▷ $u_i$ and $u_j$ are similar.
4:     else
5:         return FALSE                   ▷ $u_i$ and $u_j$ are not similar.
```

In other words, we focus on giving semantic links between user stories. We assume that T is previously defined through software engineering traceability tools. Therefore, by logical consequence, our contribution identifies Q for US.

III. RESEARCH METHODOLOGY

The goal of this study is to construct and validate a taxonomy for adding link semantics between USs, focusing on easing the task of identifying similar ones. We assume that software

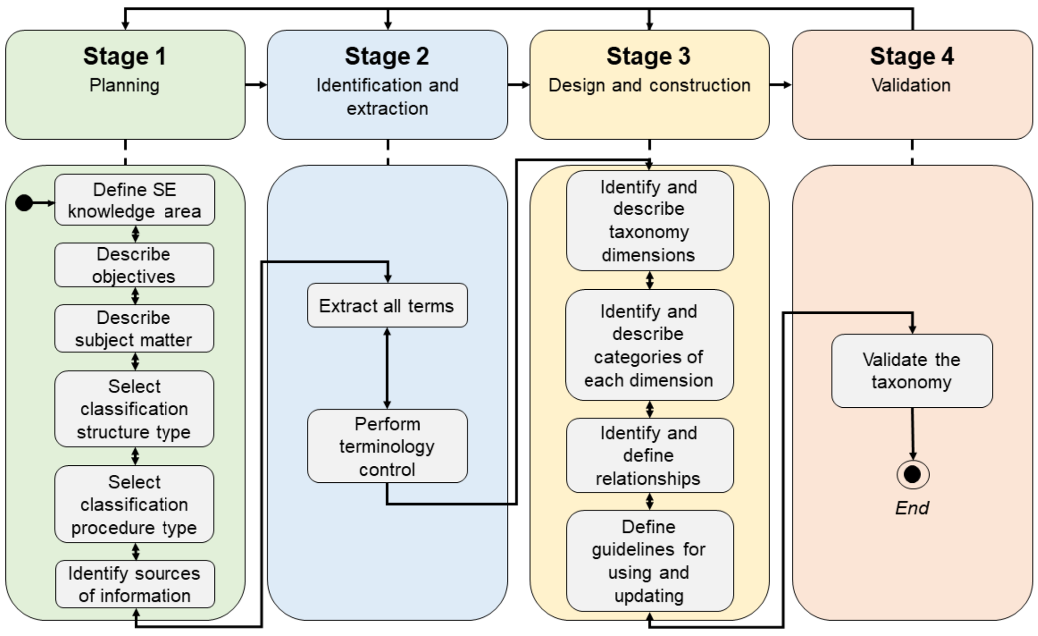

Stage 1
Planning
Stage 2
Identification and extraction
Stage 3
Design and construction
Stage 4
Validation
Define SE knowledge area
Describe objectives
Describe subject matter
Select classification structure type
Select classification procedure type
Identify sources of information
Extract all terms
Perform terminology control
Identify and describe taxonomy dimensions
Identify and describe categories of each dimension
Identify and define relationships
Define guidelines for using and updating
Validate the taxonomy
End

FIGURE 2. Employed taxonomy construction process.

artifacts derive from the product requirements and that the requirements are documented as USs. Therefore, by having traceability between the software requirements and other software artifacts (e.g., tasks, test cases, and commits), we can reuse the assets by calculating the similarity between USs (see Algorithm 1), in other words, the USs serve as a reference for a reuse mechanism such as information retrieval or recommender systems. Given this, we formulated the following research questions:

- **RQ1**: How to construct a taxonomy for USs of WIS to enable the reuse of US-related assets?
- **RQ2**: Does the proposed taxonomy enable the reuse of US-related assets?

To answer the research questions, we employed the taxonomy development method presented by [27]. The executed process is summarized in Figure 2 and further described as follows. The research questions are further defined and refined in Section V.

A. PLANNING

For the planning phase, first, we defined the software engineering knowledge area related to our taxonomy according to the Software Engineering Body of Knowledge (SWEBOK) [29] as *software requirements*. Afterward, we defined the main objective of the proposed taxonomy as to *define a set of categories to classify the USs of WIS in a level of granularity that enables the reuse of US-related assets*. Since US is the most popular notation for requirement specification in ASD, the subject matter to be classified are USs.

For the structure of the taxonomy, it is well-known that facet-based classification is adequate to organize reusable software development assets [25], [26] given their flexibility for expansion and maintenance and ease to implement as a relational database. An example of a facet-based classification for US could be to use the dimensions of the IEEE Std. 1233: priority, criticality, risk, source, and type. We hypothesize that only considering the "type" dimension enables the reuse of requirements-related assets, such as task cards. Given this, we decided to use a facet-based classification considering only the "type" facet and refining it by detailing its arrays (or sub-facets) [30].

The procedure to classify the user stories was qualitative and based on thematic analysis. Finally, the basis (i.e., sources of information) to define the taxonomy were product backlogs from real-world projects.

B. IDENTIFICATION AND EXTRACTION, AND DESIGN AND CONSTRUCTION

We executed stages two ("Identification and extraction") and three ("Design and construction") by applying a thematic analysis procedure on five product backlogs composed of a total of 118 USs. For the thematic analysis procedure, we executed the five steps recommended by Cruzes and Dybå [31]: (i) data extraction, (ii) data coding, (iii) translation of code into themes, (iv) creation of a model of higher-order themes, and (v) trustworthy assessment of the synthesis.

For the first step (i) of the employed thematic analysis procedure, one researcher extracted the product backlog items of the five product backlogs used as the information source to a spreadsheet, containing a tab to store the items of each product backlog. Since the extraction was executed manually, for reliability purposes, another researcher checked if the extraction was correct. Afterward (step ii), each product backlog was analyzed by a researcher (i.e., data extractor), who coded the product backlog items. For this purpose, the data extractor labeled and coded segments of the product backlog items given attributes that he judged to be relevant such as feature, requirement, and system interface. To reduce researcher bias, a second researcher (i.e., data checker) reviewed the labels and codes defined for each product backlog, and in case of conflict, all the involved researchers discussed it during a specific meeting.

To define the themes (step iii), the researchers, during workshops, analyzed the codes for all included USs. At the end of this process, the themes were identified and used as input for step (iv), in which the taxonomy was defined by relating the identified themes and sub-themes, as necessary. The last step (v) was executed as part of the taxonomy validation process (see Section III-C). The resulting taxonomy and its usage guideline are presented in Section IV.

C. VALIDATION

Existing validation approaches for a taxonomy include demonstrating the orthogonality of its dimensions, benchmarking against existing classifications and by demonstrating its utility to classify existing knowledge [27], [32], [33]. The orthogonality, which refers to the mutually exclusive nature of the classification, is implied by its structure, and we discuss it in Section IV-A. We demonstrate its utility in terms of its representativeness, capability to enable the reuse of US-related assets (i.e., specifically, task cards), and the practical utility. For this purpose, we followed a two-steps approach: an offline study and an online study (i.e., case study). For the simulation, we collected 530 USs and 1879 tasks from 26 ended projects of four software development companies. For the case study, we applied the taxonomy into two ongoing projects for three months, which resulted in analyzing 35 USs and 192 tasks, and involving nine software developers. We present the details regarding the validation procedure's planning and results in Section V.

IV. PROPOSED SOLUTION

This section presents the proposed taxonomy for US, tailored to WIS, and its usage guideline. The terms of the taxonomy represent the set of classification terms C, as discussed in Section II. In what follows, Section IV-A presents the proposed taxonomy, and Section IV-B discusses how to apply the proposed taxonomy to reuse US-related assets.

A. TAXONOMY OVERVIEW

As presented in Section III-B, to construct the taxonomy, we executed a procedure based on thematic analysis with five product backlogs composed of a total of 118 USs as the source of information. In what follows, we present the main decisions that we took that led us to the taxonomy shown in Table 1.

Initially, we filtered the product backlog items to be analyzed, by removing the ones that were not related to functional requirements, since they were out of the scope of this study. For instance, we removed product backlog items related to technical debt (e.g., refactor a component) or non-functional requirements (e.g., add support to internationalization and localization).

TABLE 1. Taxonomy to classify user stories.

Module	Operation
Authentication	Perform login with username and password Perform login with OAuth Password recovery First login Validate user permissions Update profile Create account Remove account
Registration	Retrieve data Update data Insert data Remove data Change data insertion
Management	Visualize Dashboard Export report to PDF Export report do XLS Notify through application Notify by e-mail

After analyzing the labels and codes, which were defined by analyzing the text of the five product backlogs after the filtering, we clustered the USs given their related modules. Specifically, we identified three modules: *Registration*, consisting of Create, Read, Update and Delete (CRUD) operations; *Authentication*, consisting of software authentication and authorization operations; and *Management*, consisting of dashboard and reports operations. Afterward, for each cluster of US (i.e., modules), we further refined the classification by identifying the main operations implemented. As a result, as shown in Table 1, the taxonomy was composed of two sub-facets: *Module* and *Operation*. It is important to notice that there is a strong inter-facet relationship between *Module* and *Operation*. For instance, the operation "First login" is only valid in the context of the module "Authentication". We can think of the modules as being mutually exclusive sets; and the operations as the elements of each set.

It is worthy to notice that the constraints related to the scope of a US (i.e., and consequently, its size) is team-dependent, not product-dependent. For instance, a team might place a constraint on the sizes of the US for their project as no more than ten days of development work [11]. Notice that, a variation between teams might be the number of days itself used as the constraint. However, more importantly, the scope is also in function of the teams' velocity. Therefore, in light of the proposed taxonomy, it is possible that a US englobes multiple operations. For instance, to deliver the same feature, team A might use one US, which is classified with [Module: "Registration", Operation: ["Retrieve data","Update data"]], and team B might use two USs, one classified as [Module: "Registration", Operation: "Retrieve data"] and another classified as [Module: "Registration", Operation: "Update data"]. Even though this characteristic hinders the orthogonality of the classification, it does not hinder the reuse of assets (e.g., information retrieval or recommendation system), which is the proposed taxonomy's goal.

B. GUIDELINES TO USE THE PROPOSED TAXONOMY

The process to classify a US starts when the user registers a new US into the database. If the US is related to functional requirements, in addition to the basic US information (i.e., title, description, etc.), the user must also select a module and the operations that represent the User Story. In terms of operationalization, the module and operations might be represented as tags. Figure 3 shows the steps to classify a US using the defined taxonomy and presents an example in which the US was classified as [Module: "Authentication", Operation:"Login"]. Such a classification refers to the mapping M of user stories to C, as discussed in Section II.

In the cases in which there is no valid classification for a US, a new module or operation might be created. In this case, we recommend that a Maintenance Group should be responsible for the extension of the taxonomy. The Maintenance Group should also audit the classifications, to ensure correctness and avoid corrupting the database. Once the USs are categorized, it is possible to retrieve them as well as their related assets based on their module and operation. Figure 4 shows the retrieval of all USs in a database classified with *Authentication* module and *Login* operation.

An advantage of using the proposed taxonomy is that it enables the improvement of the traceability between US-related assets. To demonstrate this, we consider the case for task cards. Using a similar approach as the one shown in Section III to classify the USs and the same database (i.e., set of US and the related task cards), we classified 336 task cards into "Task type". For each "Operation", a set of tasks were defined. As with the inter-facet relationship of the facets "Module" and "Operation" of the entity US, there is a strong relationship between the facet "Task type" of the entity task card and the facet "Operation" of the entity US. Table 2 shows the number of defined tasks for each US module and operation while Table 3 shows an example of tasks defined to be reused for the *Registration* module and *Insert data* operation.

Having a taxonomy for the task cards increases the level of granularity of the traceability of USs and task cards (see Figure 5) and, consequently, increases the reuse capabilities. The reasoning for this conclusion follows from the fact that with a higher level of granularity, it is possible to improve the characterization of the given entities (i.e., USs and task cards), therefore, having a more precise means to calculate similarity. Thus, based on the US's module and operation, the user can retrieve (i.e., reuse) the standard tasks associated with it. As a consequence, this increases the potential of reuse of software engineering assets. For instance, assuming that the task cards are classified given our taxonomy and that they are linked to other assets such as commits, source code, or documentation, we enable the user to retrieve this information efficiently.

TABLE 2. Amount of defined base tasks by module and operation.

	Operation	Tasks
Authentication	Perform login with username and password	13
	Perform login with OAuth	3
	Password recovery	6
	First login	8
	Validate user permissions	5
	Update profile	7
	Create account	8
	Remove account	2
Subtotal		52
Registration	Retrieve data	14
	Update data	8
	Insert data	9
	Remove data	6
	Change data insertion	9
Subtotal		46
Management	Visualize Dashboard	4
	Export report to PDF	9
	Export report do XLS	6
	Notify through application	7
	Notify by e-mail	7
Subtotal		33
Total		131

TABLE 3. Examples of previously defined tasks for registration module and insert data operation.

Module	Operation	Task
Registration	Insert data	Create insertion screen Create database entity Create data insertion route Validate client side data Make insertion screen call Create insertion method Validate server side data Consume insertion service Persist data on database

V. VALIDATION OF THE TAXONOMY

This section details the validation procedure applied to the proposed taxonomy. The procedure consists of two steps: an offline study (Section V-A) and an online study (i.e., a case study) (Section V-B). Moreover, the procedure addresses **RQ2**, defined in Section III.

The objective of **RQ2** is to assess the taxonomy's potential to enable reusing USs and related artifacts. For this purpose, we evaluated **RQ2** given three perspectives: (i) reuse of US and (ii) reuse of related artifacts, and (iii) practical utility. First, we analyzed the taxonomy's potential to represent the USs related to functional requirements (i.e., business USs) in the context of WIS. Thus, we further refined **RQ2** into:

- **RQ2.1:** What is the proposed taxonomy's potential for reusing USs?
- **RQ2.2:** What is the proposed taxonomy's potential for reusing USs related artifacts?

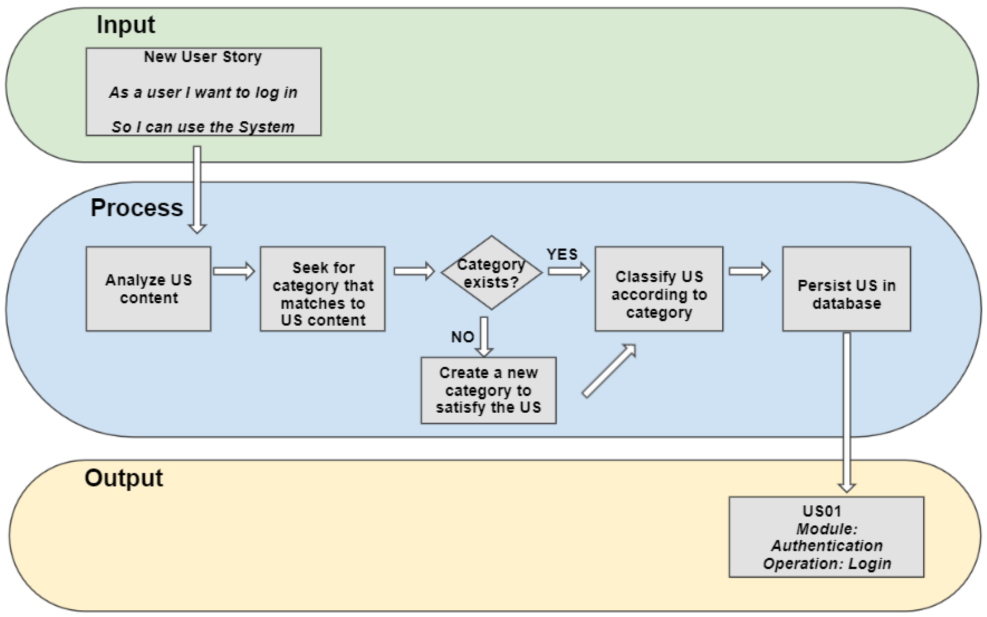


FIGURE 3. Process to classify USs based on the proposed taxonomy.

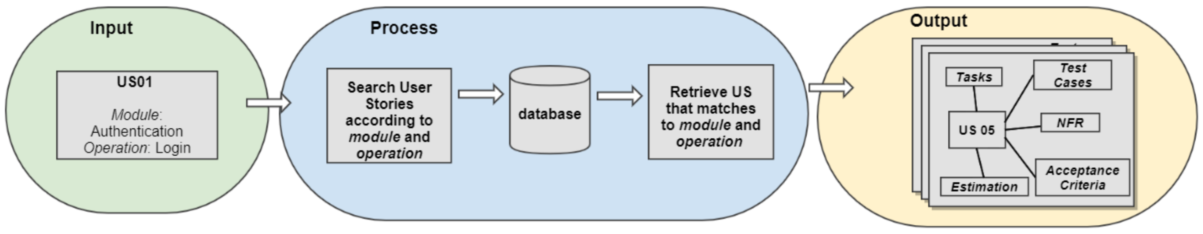


FIGURE 4. The US taxonomy allows the retrieval of USs and their related assets.

- **RQ2.3:** What is the practical utility of using the proposed taxonomy for reusing USs and related artifacts?

For **RQ2.1**, since the taxonomy represents the terms to be used for classifying WIS user stories, in other words, it defines the elements of C (see Section II), we evaluated it given the following mapping properties of M: coverage, granularity, and heterogeneity. The problem associated with this research question is analogous to the classical model fitting problem of machine learning. Our goal is that the taxonomy can classify the highest number possible of USs. Conversely, since we intend to use the taxonomy to classify USs, using the results as a reference to enable calculating similarity between them, if we use too few taxonomy terms for classification (i.e., underfitting) our too many (i.e., overfitting), it will not be useful.

For instance, let's consider two extreme cases. Consider that we have 100 USs, and that, theoretically, we have 20 types of USs uniformly distributed. In other words, as expected values for the mapping properties, we have:

- Expected **coverage**: 100%.
- Expected **granularity**: 20%.
- Expected **heterogeneity**: uniformly distributed over the 20 terms used to classify the user stories.
- Expected **complexity**: 20%.

If by applying the taxonomy to classify all the USs (i.e., calculated **coverage** of 100%) we only have 1 term used (i.e., calculated **granularity** of 1), the classification is useless, because it follows that the USs are all similar (i.e., a lot of false-positive). Conversely, if we need 100 (i.e., calculated **granularity** of 1) taxonomy terms to classify them, it is also useless, because it follows that all USs are different (i.e., a lot of false-negative). Additionally, even if we use 20% (i.e., calculated **granularity** of 1, as expected), but the distribution of the used terms to classify the user stories do not follow the expected **heterogeneity**, we might have false-positives and false-negatives. Finally, even if all the expected values are achieved, there is still the risk of having false-positives and false-negatives. Therefore, we conclude that the metrics serve as a filter to discard invalid classification schemes, but not to confirm them as valid. For confirmation purposes, there is the need for human domain expertise to analyze if each user story was classified correctly following the available taxonomy

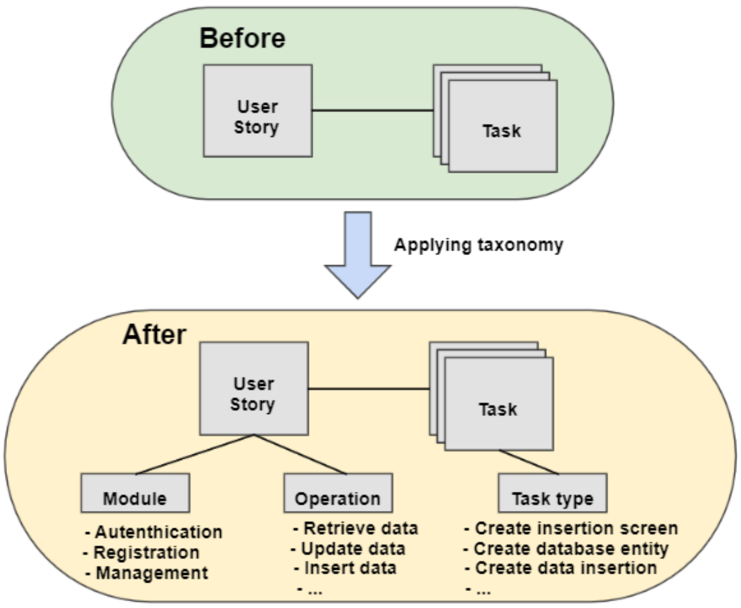


FIGURE 5. Increase of the granularity of traceability between US and task cards after applying the proposed taxonomy.

terms. Finally, the **complexity** indicates the potential level of reuse of the user stories. Given that the classification is correct, the lower the complexity, the higher is the degree of reuse of user stories.

Unfortunately, for a quantitative taxonomy fit analysis, we would need to know real **granularity** and **heterogeneity**, which is not available, because this problem is an instance of the classical *Chicken and Egg Problem*. Despite this, due to the nature of WIS, we expected to classify over 85% of the business User Stories, leaving some margin for unexpected features for pure WIS such as product recommendation, using 3 modules (i.e., Registration, Authentication, and Management), having most of the US's (over 70%) to be classified as of *Registration*, and the remaining ones distributed between *Authentication* and *Management*.

RQ2.2 aims to assess if the proposed taxonomy enables the reuse of US-related assets. We restricted our evaluation for this research question to reusing task cards. For this purpose, we defined a metric called *Task Reuse*, which is calculated using Equation 3.

$$Tasks\ Reuse = \frac{Number\ of\ Reused\ Task\ Cards}{Total\ Number\ of\ Task\ Cards} * 100 \quad (3)$$

RQ2.3 concerns the evaluation of the practical utility of the taxonomy, focusing on the reuse of task cards from the perspective of software engineers. To this end, we applied the TAM. It is essential to notice that, as discussed in Section II, our contribution is limited to defining C, the set of terms to classify user stories. Despite this, to be able to evaluate the taxonomy from the viewpoint of software engineers, we defined a manual procedure, based on the guidelines presented in Section IV-B, for M. Therefore, even though, by applying TAM we implicitly assess C and M, our focus is on C. However, the data collected may serve as indicators for future work in refining M.

A. OFFLINE STUDY

The goal of the offline study was to investigate the feasibility and potential limitations of the proposed taxonomy. In this step, we collected data from 26 product backlogs from delivered projects of four software development companies. The projects were composed of 530 USs and 1879 task cards. Then, we applied the process illustrated in Figure 3 to classify the product backlogs.

The classification was performed (manually) using a spreadsheet to register the data. Each product backlog was distributed to a researcher, and it was filtered by removing the USs not related to functional requirements. As a result, 123 USs were discarded, remaining in 407 USs to be classified. We followed the same peer review process used to construct the first version of the taxonomy, discussed in Section IV. Afterward, each researcher classified his assigned USs by labeling them with the proper modules and operations, according to the inter-facet relationship discussed in Section IV-A. These steps were executed by four researchers following peer audits, seeking agreement among researchers, to avoid researcher bias [34].

None of the USs were classified using multiple modules or operations since it was not necessary. Whenever the researcher could not classify a US given its scope or taxonomy limitation, it was labeled as *Not covered*. It is worth to mention that a risk identified during the classification process was the possibility that researchers could not understand the context of a US based on its description (e.g., a US described as ‘‘Implement feature’’) since the US is an artifact intended

TABLE 4. A sample of user stories classified during the offline study.

Project	#	US description	Module	Operation
X	US1	Login on system	Authentication	Do login with username and password
X	US2	Register rooms	Registration	Insert data
X	US3	Create remote service to validate information	*Not Covered*	*Not Covered*
Y	US4	Register transactions	Registration	Insert data
Y	US5	Update transactions	Registration	Update data
Z	US6	Generate infrastructure report	Management	Export report to PDF
Z	US7	Database refactoring	*N/A*	*N/A*
Z	US8	Develop feature	*Not Covered*	*Not Covered*

to be used locally by the project team. To improve the reliability of the validation, whenever the description of the US was not enough information, we also checked its related task cards because they could provide more information regarding the US scope.

Finally, to assess the indexing potential of reusable assets (**RQ2.1**), the researchers evaluated the USs-related task cards. Each of them evaluated only the tasks related to the USs previously assigned to him, using the scheme presented in Section IV-B. We assumed that every task classified can be reused since they can be retrieved based on their "Task type".

1) OFFLINE STUDY RESULTS

This section outlines the results of the offline study. Table 4 shows examples of USs from three different product backlogs (i.e., X, Y, and Z) and their assigned modules and operations. Despite belonging to different projects, US2 and US4 can be considered similar since they were classified with the same module and operation. US7 is an example of a user story removed of the project's product backlog because it is not related to a functional requirement. On the other hand, US3 is an example of a US not covered by the taxonomy. US8 illustrates an example of a user story that lacks relevant information in its description, which represents a challenge in the classification process, as discussed in Section V-A. In these cases, it is necessary to analyze the task cards of the corresponding user story to support the process. However, for US8, even a further investigation into its tasks' content has not revealed its scope.

a: POTENTIAL TO REUSE USER STORIES (RQ2.1)

As mentioned before, we evaluated the taxonomy's potential to enable the reuse of user stories in terms of its coverage (M_c), granularity (M_g), and heterogeneity (M_h). Out of the 407 USs, we classified 367 with the proposed taxonomy, achieving coverage (M_c) of 90.17%, above our expected coverage of 85%. We classified the US using all the 18 operations of the taxonomy, therefore, having a granularity (M_g) of 18. Regarding the heterogeneity (M_h), 80.11% were classified as *Registration*, 10.35% as *Authentication*, and 9.54% as *Management*, following the expected heterogeneity. Further, the mapping properties (i.e., M_c, M_g, M_h) and the mapping

TABLE 5. Sample of classified tasks of the taxonomy adequacy step.

Project	#	US desc.	Task desc.	Module	Operation	Task type
W	US1	Create contacts	Create insertion screen for contact creation	Registration	Insert data	Create screen
W	US1	Create contacts	Create method to validate contact data	Registration	Insert data	Validate cliente side data
Z	US22	Create issues	Validate user permissions	Registration	Insert data	N/A
Z	US22	Create issues	Create DAO method for issue insertion	Registration	Insert data	Persist data on database

itself were analyzed by a researcher who has not participated in the classification process but had previous knowledge about the taxonomy. As an outcome, the results were judged to be satisfactory. Finally, we had a calculated complexity (M_x) of 4.9%, indicating a high potential of user story reuse, since we can represent 367 with only 18 types.

b: POTENTIAL TO REUSE USs RELATED ARTIFACTS (RQ2.2)

We evaluated the reuse potential of the proposed taxonomy by analyzing the reuse of task cards (Equation 3). To accomplish that, we applied the taxonomy to a set of task cards related to the previously classified USs. From the initial set of 1879 task cards, we removed 673, resulting in 1206 to be classified.

Table 5 shows a sample of tasks of two projects (W and Z). The first two lines show two cases of project W in which the tasks were classified using a *task type* related to a tuple of module and operation. The third line shows an example of a task covered by the taxonomy. The last line shows a classified task from project Z. Considering the 1206 task cards, the taxonomy achieved 80.93% of reuse, resulting in 976 classified task cards. This percentage indicates the reuse potential of the taxonomy.

B. CASE STUDY

According to Runeson and H́ost [35], a case study is an empirical method to investigate a contemporary phenomenon in a specific context. In this study, the investigated phenomenon consists of using the proposed taxonomy to classify USs and, hence, to enable the reuse of task cards. The main goal of the case study is to evaluate the cost-benefit of the taxonomy from the perspective of development teams regarding the reuse of user stories and task cards.

To accomplish that, we conducted an embedded case study in a small software company, which uses Scrum to manage its projects. Overall, the company has 26 employees performing different functions such as project manager, quality analyst, and developer. The teams are composed of four to five members working on the development of Web enterprise applications using several technologies such as Java, AngularJS, NodeJS, and MongoDB. The most experienced developer, in each team, acts as the Scrum master, conducting planning, review, and retrospective meetings. In contrast, the product owner works in the same company and serves as a representative of the customer.

The case study addressed the research questions **RQ2.1**, **RQ2.2**, and **RQ2.3**. To answer them, we considered two ongoing projects of the previously mentioned company as research units. One of the projects was composed of five developers, whereas the other one was composed of four developers. Thus, we considered nine research subjects during the case study. This step of the research lasted three months. Firstly, the researchers executed two workshops of one hour to train the software developers on how to use the taxonomy. Then, the software developers started using the taxonomy during the Sprint planning meetings through a spreadsheet prepared by the researchers. During the execution of the case study, the researchers acted as the Maintenance Group, auditing the

To address **RQ2.1** and **RQ2.2**, we followed an approach similar to the one applied during the offline study (see Section V-A), in which the process illustrated in Figure 3 was executed by the software developers. At the end of the period, the developers have described 35 user stories and 192 task cards. Based on these data, the researchers gathered the metrics presented in Section V.

On the other hand, to address **RQ2.3**, we applied the Technology Acceptance Model [28], which is widely used to assess the practical utility of a proposed solution [36], [37]. TAM considers that the adoption of any technology is influenced by its perceived usefulness (PU) and its perceived ease of use (PE). PU refers to the degree to which an individual believes that using a particular technology would enhance his or her job performance. On the other hand, PE refers to the degree to which an individual believes that using a specific technology would be free of physical and mental effort [38].

We operationalized TAM through a questionnaire composed of fourteen questions regarding the PU and fourteen regarding the PE variable [39]. For each question, we used a five-level Likert scale to collect participants' responses, ranging from 0 (Strongly disagree) to 4 (Strongly agree). Appendix A presents the applied questionnaire. At the end of the period of the case study, the nine software developers (subjects of research) answered the questionnaire.

1) CASE STUDY RESULTS

In this section, we present the results of the case study regarding **RQ2.1**, **RQ2.2**, and **RQ2.3**.

a: POTENTIAL TO REUSE USER STORIES (RQ2.1)

As with for the offline study, we evaluated the taxonomy's potential to enable the reuse of user stories in terms of its coverage (M_c), granularity (M_g), heterogeneity (M_h), and complexity (M_x). From the 63 user stories observed during the period of the case study, 59 were related to functional requirements (88%). In this case, we reached coverage (M_c) of 100%. The software developers used 6 operations; therefore, we had a granularity (M_g) of 6. Furthermore, regarding the heterogeneity (M_h), 89.8% were classified as *Registration*, 9.4% as *Authentication*, and 0.2% as *Management*. Finally, the researchers analyzed the classification performed by the software developers and judged them as satisfactory. With regards to the complexity (M_x), we had a calculated M_x of 10%.

TABLE 6. US distribution over modules and operations.

Module	Operation	Tasks reused
Authentication	Perform login with OAuth	3
	Validate user permissions	5
Subtotal		8
Registration	Retrieve data	9
	Update data	7
	Insert data	11
	Change data insertion	13
Subtotal		40
Total		48

TABLE 7. US distribution over modules and operations.

Module	Operation	User Stories	Tasks
Authentication	Perform login with OAuth	1	4
	Validate user permissions	1	6
Subtotal		2	10
Registration	Retrieve data	8	30
	Update data	3	8
	Insert data	5	33
	Change data insertion	13	55
Subtotal		29	126
Total		31	136

b: POTENTIAL TO REUSE USs RELATED ARTIFACTS (RQ2.2)

As with for the offline study, we evaluated the taxonomy's potential to enable the reuse of USs related tasks in terms of the reused task cards. In total, the software developers specified 192 task cards in the two projects' product backlogs, of which 142 stemming from business user stories. Of this amount, they classified 136 using the proposed taxonomy, i.e., a reuse rate of 95.77% (see Table 7). On the other hand, they did not use the taxonomy in 4.22% of all cases, i.e., 6 tasks only. These task cards, not reused, were added to the task database for future reuse.

Table 6 shows the number of tasks by *Operation* specified in both projects. As can be seen, most of the reused task cards (40 tasks or 85.3%) are related to the *Registration* module. We expected these results since both projects were developing Web Information Systems.

Although the proposed taxonomy includes 131 types of task cards, the software developers used just 48 of them to classify all their projects' tasks during the period of the case study. Table 7 presents the number of task cards reused by module and operation. As expected, in most cases, the developers reused tasks of the "Registration" module.

c: PRACTICAL UTILITY ASSESSMENT (RQ2.3)

We assessed the taxonomy's practical utility by evaluating the answers to a questionnaire based on TAM to assess the practical utility of the proposed taxonomy for user stories. In total,

TABLE 8. Results for perceived usefulness from TAM.

ID	Item	Agreement	Neutral	Disagreement
PU1	Job Difficult Without	**5**	1	3
PU2	Control Over Work	**9**	0	0
PU3	Job Performance	**5**	3	1
PU4	Addresses My Needs	**7**	2	0
PU5	Saves Me Time	**4**	2	3
PU6	Work More Quickly	3	3	3
PU7	Critical to My Job	**5**	2	2
PU8	Accomplish More Work	2	4	3
PU9	Cut Unproductive Time	**5**	4	0
PU10	Effectiveness	**5**	3	1
PU11	Quality of Work	**7**	1	1
PU12	Increase Productivity	4	4	1
PU13	Makes Job Easier	**7**	1	1
PU14	Useful	**8**	1	0

TABLE 9. Results for perceived ease of use from TAM.

ID	Item	Agreement	Neutral	Disagreement
PE1	Confusing	1	1	**7**
PE2	Error Prone	0	2	**7**
PE3	Frustrating	1	2	**6**
PE4	Dependence on Manual	1	1	**7**
PE5	Mental Effort	0	1	**8**
PE6	Error Recovery	**6**	2	1
PE7	Rigid and Inflexible	0	2	**7**
PE8	Controllable	**7**	2	0
PE9	Unexpected Behavior	0	1	**8**
PE10	Cumbersome	0	1	**8**
PE11	Understandable	**7**	2	0
PE12	Ease of Remembering	**8**	1	0
PE13	Provides Guidance	**6**	3	0
PE14	Easy to Use	**8**	1	0

we gathered nine questionnaire responses, i.e., one for each software developer. We summarize the results regarding the perceived usefulness and the perceived ease of use of the proposed taxonomy in Tables 8 and 9, respectively. For example, in question PU1 (Table 8), we asked the software developers if their job is more difficult without using the taxonomy for user stories. As a result, five subjects agreed (i.e., they answered "agree" or "strongly agree'), one subject neither agreed nor disagreed, and three subjects disagreed (i.e., they answered "disagree" or "strongly disagree') that their job can be harder without using it.

In Appendix A, Tables 10 and 11 present the questions regarding the perceived usefulness and perceived ease of use, respectively.

VI. DISCUSSION

This section discusses the obtained results in light of the research questions **RQ2.1**, **RQ2.2**, and **RQ2.3**. Moreover, we examine the implications of the achieved results for academia and industry.

A. RQ2.1: WHAT IS THE PROPOSED TAXONOMY'S POTENTIAL FOR REUSING USs?

The proposed taxonomy includes three modules (Authentication, Registration, and Management) and 18 operations. During the offline study, the taxonomy was able to cover (M_c) 90.17% of the business USs present in the 26 product backlogs and for the case study, 100%. Regarding the granularity (M_g), the offline study recorded 18, and the case study, 6. The case study used fewer taxonomy terms (i.e., operations) due to the lower number of user stories and their similarity. Therefore, for this case, it was expected to have a lower calculated granularity, when compared to the offline study. The calculated heterogeneity (M_h) for the offline study followed the expected distribution, but for the case study not. By analyzing these results, we concluded that it was due to the nature of the products being developed. Therefore, we have concluded that the results do not indicate the need to refine the taxonomy.

Combining all the user stories used to construct and validate the taxonomy, we have a total of 707 Business USs, in which 663 were classified using only 18 terms, resulting in an overall complexity (M_x of 2.7%, demonstrating the high potential of reusing user stories. Therefore, we observed that our initial hypothesis that reusing USs is a valid opportunity to be explored given its potential to increase software development efficiency. Further, our results demonstrate that using

TABLE 10. TAM - perceived usefulness questions.

ID	Measure Instrument
PU1	My job would be difficult to perform without the approach.
PU2	Using the approach gives me greater control over my work.
PU3	Using the approach improves my job performance.
PU4	The approach addresses my job-related needs.
PU5	Using the approach saves me time.
PU6	The approach enables me to accomplish tasks more quickly.
PU7	The approach supports critical aspects of my job.
PU8	Using the approach allows me to accomplish more work than would otherwise be possible.
PU9	Using the approach reduces the time I spend on unproductive activities.
PU10	Using the approach enhances my effectiveness on the job.
PU11	Using the approach improves the quality of the work I do.
PU12	Using the approach increases my productivity.
PU13	Using the approach makes it easier to do my job.
PU14	Overall, I find the approach useful in my job.

TABLE 11. TAM - perceived ease of use questions.

ID	Measure Instrument
PE1	I often become confused when I use the approach.
PE2	I make errors frequently when using the approach.
PE3	Interacting with the approach system is often frustrating.
PE4	I need to consult the user manual often when using the approach.
PE5	Interacting with the approach requires a lot of my mental effort.
PE6	I find it easy to recover from errors encountered while using the approach.
PE7	The approach is rigid and inflexible to interact with.
PE8	I find it easy to get the approach to do what I want it to do.
PE9	The approach often behaves in unexpected ways.
PE10	I find it cumbersome, to use the approach.
PE11	My interaction with the approach is easy for me to understand.
PE12	It is easy for me to remember how to perform tasks using the approach.
PE13	The approach provides helpful guidance in performing tasks.
PE14	Overall, I find the approach easy to use.

taxonomy as the means to add link semantics to user stories is a promising approach to enable reusing them. Although the taxonomy is tailored to a specific type of system (i.e., WIS) and, naturally, the results can not be generalized, we believe that they are promising.

B. RQ2.2: WHAT IS THE PROPOSED TAXONOMY'S POTENTIAL FOR REUSING USs RELATED ARTIFACTS?

In the offline study, we observed a reuse rate of 80.93% of task cards, which means that 976 task cards were reused. On the other hand, in the case study, we achieved an improved reuse rate of 95.77% of the task cards. In this case, 48 distinct task types enabled the reuse of 136 of 142 task cards specified by the software developers during the case study period. Therefore, there is evidence that the taxonomy can enable the reuse of user story-related assets, which can improve the efficiency of software development processes.

C. RQ2.3: WHAT IS THE PRACTICAL UTILITY OF USING THE PROPOSED TAXONOMY FOR REUSING USs AND RELATED ARTIFACTS?

To answer RQ2.3, we analyzed the results summarized in Tables 8 (PU) and 9 (PE), in which positive responses are highlighted in bold.

Based on the data of Table 8, we verify that the research participants indicated good acceptance for 11 of 14 items of PU, since only items PU6, PU8, and PU12 did not receive mostly positive evaluations. Overall, the subjects of research agreed to the usefulness of the proposed taxonomy in 76 of 126 possible responses regarding PU variable, i.e., 60% of all individual responses. In contrast, they assessed it negatively in only 19 of 126 responses, i.e., 15% of all individual responses. Therefore, it is possible to state there is evidence that the software developers considered the proposed taxonomy for user stories useful.

Regarding the responses to PU6, PU8, and PU12, we believe that they are evidence of the need to define a systematic procedure to apply the taxonomy, possibly needing automation and reducing the effort from the developers to use it. Further, notice that, in the case study, the developers have not experienced the potential of artifacts reuse, such as test cases, non-functional requirements, risks, and so on. Therefore, their perspective regarding the potential gain on productivity given the reuse of artifacts is limited. The taxonomy itself is not enough to increase the developers' productivity, but the reuse enabled by it, yes. Therefore, such responses were expected and did not hinder our conclusions regarding the taxonomy's usefulness.

On the other hand, by observing the data of Table 9, we verify the research subjects indicated an affirmative acceptance for all 14 items regarding the perceived ease of use from TAM. Overall, the participants of the research agreed that the proposed taxonomy is efficiently handling in 100 of 126 possible responses regarding PE variable, i.e., an acceptance of the perceived ease of use of 79,3%. In contrast, they assessed it negatively in only 4 of 126 responses, i.e., 3,1% of all individual responses. Therefore, it is possible to state there is evidence that the software developers considered the taxonomy for user stories easy to use.

Thus, since most of the subjects of the case study expressed affirmative acceptance for both variables of TAM, we concluded that there is evidence of the practical utility of the taxonomy as a means for giving link semantics to USs. Therefore, it is a promising approach for enabling the reuse of agile software development assets.

D. IMPLICATIONS FOR ACADEMIA

The taxonomy can be used by researchers to mitigate several SE problems. For instance, the team multiple formation problem consists of the allocation of multiple individuals, with different sets of skills, into multiple projects, with different requirements, to maximize the matching between them. An essential step of the team formation process is to build reliable developer profiles. Task cards may be promising assets to build these profiles since they may represent the developers' technical skills. However, traditionally, task cards are written using nonstandard text descriptions, which hidden to determine the similarity between them. For example, the tasks ''Create login page'', ''Build authentication interface'', and ''Developer login interface'' have the same purpose, but they are represented with different descriptions. Hence, it is difficult for computational techniques to recognize them as single task cards. The proposed taxonomy enables the reuse of task cards, even those from different product backlogs, to build more reliable developer profiles.

The taxonomy can also be used to support non-functional requirements (NFRs) recommendation. NFRs define constraints or restrictions on the design of the system, related to performance, maintainability, usability, scalability, and others. NFRs are linked to functional requirements. However, Product Owners usually focus on the latter and tend to neglect the previous ones. Since the classified USs and their related assets are traceable, researchers can use specialized techniques to improve the recommendation of NFRs.

Another Common SE problem is the effort estimation problem. Effort estimation is the process of predicting the effort required to develop a requirement. The literature presents some studies proposing effort estimation methods based on historical data and machine learning techniques [40]. These techniques use historical data to train and provide more accurate effort estimates. In this case, the taxonomy can be useful since it enables us to retrieve similar USs, already estimated in the past, and use them for training the machine learning techniques.

Our focus in this study was on the taxonomy itself as a classification scheme, and not in the procedure to apply it for classifying user stories. Therefore, an improvement opportunity is to develop methods to help practitioners to classify the USs and task cards, using the taxonomy as the basis. For this purpose, decision-support tools, such as chatbots, can be developed to assist in the classification process. Another possibility is to use a dataset of user stories previously classified to train an ontology by using Natural Language Process techniques.

The proposed taxonomy is tailored to WIS type; however, we encourage researchers to conduct more empirical studies on other domains to evolve the taxonomy. New modules and operations can be identified and may increase the representativeness and potential of reuse of the taxonomy.

E. IMPLICATIONS FOR INDUSTRY

TThe USs are the main assets to capture descriptions of software features in ADS and to facilitate product-related discussions between the stakeholders. The proposed taxonomy allows the USs to became central points to establish traceability links between themselves and their related assets. Therefore, the results presented in this article can support practitioners to improve the efficiency of software planning activities. For instance, assuming the taxonomy has been applied for a sufficient period in a software development company that uses the Scrum framework to manage their projects. The taxonomy can assist the Scrum teams in decomposing USs into task cards during the sprint planning meetings. The team can retrieve similar USs from past projects and analyze their task cards before performing the break down of current USs. Similar USs have a higher probability of having more task cards in common. This support can be specifically useful for low experienced teams that may face challenges to perform the US break down.

Similarly to the researchers, practitioners can also benefit from the taxonomy for effort estimation purposes. One commonly used method during the estimation process is the Planning Poker, which consists of assigning story points to sprint backlog items. In this case, the Scrum team can retrieve similar USs and examine their estimated value. Afterward, the team can execute the Planning Poker or another proper method. This practice can help team members to provide more accurate effort estimates since they can use baseline USs to make their decision.

Moreover, the taxonomy enables practitioners to quickly establish traceability links between USs, tasks, and related commits. Therefore, practitioners can also search for similar USs and tasks to examine the source codes from the associated commits, as done by StackOverflows' users, when they search for programming questions. An advantage of using the taxonomy is that, since the code owner is a company's employee, it may be possible to share the experience or ask questions. Therefore, the taxonomy can serve as a means to promote knowledge sharing across the organization. Additionally, project managers can explore this traceability to build project statistics and improve the decision-making process. For instance, it would be possible to track the number of bugs related to specific USs and plan actions to reduce them.

VII. THREATS TO VALIDITY

Validity threats concern factors that may have affect the outcome of our study. In this section, we discuss those threats considering categories proposed by [41]: *internal*, *construct*, and *reliability*.

A. INTERNAL VALIDITY

The process to categorize US's during the offline study was made by the distribution of backlogs among the authors of this article, which manually analyzed each US and selected the most appropriate category. This process could generate two different categories for similar U's analyzed by different authors, resulting in an inconsistent database and consequently inconsistent results. However, the categorization made by the authors was peer-reviewed to avoid a misunderstanding of the meaning of each category, resulting in a more consistent and realistic database, spreading the knowledge and avoiding researcher bias.

B. CONSTRUCT VALIDITY

The results from the case study were reported on the metrics of categorization *coverage* and *representativeness*. Although concluding that the proposed taxonomy can be used to classify and retrieve historical data based on the meaningful results – 100.00% of coverage and 18 categories to represent all the business US's – there is no empirical support that indicates the consistency of those numbers, which can be further collected in future works. Although the results from tasks reuse were also very expressive – 95.77% for tasks reuse, and 35.29% for tasks representativeness –, the same validity applies since there is no empirical support to indicate the conclusion.

C. RELIABILITY VALIDITY

The categories, as well as the tasks used in the case study, were defined based on the definition dataset from projects within one company and improved based on the improvement dataset. Additionally, the research collected a high number of backlogs from four companies. Both backlogs – the ones used to define the categories used in the case study, as well as the ones used to refine them – consisted of web information systems type, which explains the significant number of registration US's and tasks. Considering another context and the categories defined must bring different numbers, once the defined categories have a direct link with the type of the system.

VIII. CONCLUSIONS AND FUTURE WORK

This article proposed a taxonomy for user stories and evaluated its potential to promote reuse-driven software engineering for agile software development. The taxonomy is tailored to Web Information Systems, and it was validated using product backlogs from software projects of four software development companies.

The validation process was divided into two steps. First, we conduct an offline study to investigate its feasibility and potential problems. To accomplish that, we applied the taxonomy in 26 product backlogs from delivered projects, including 530 USs and 1879 task cards. Afterward, we performed a case study in two projects, containing 35 USs and 192 task cards. The results indicate that the taxonomy enables the reuse USs and their related assets. Besides, we applied the Technology Acceptance Model, which allowed us to evaluate the taxonomy's ease of use and its perceived usefulness. The results pointed out that teams readily adopted the taxonomy.

The contribution of this study can be further improved and opens a new research field that explores the extensive reuse of software artifacts in agile projects, even among different projects, as well as the possibility of adding intelligent algorithms to aid software developers with past projects experience proactively.

As a limitation, our current solution focused on proposing the classification scheme (i.e., taxonomy) for enabling adding link semantics between user stories, and, consequently, promoting agile artifacts reuse. Conversely, in this study, the researchers and software engineers manually classified the USs given the proposed taxonomy. Although we recorded positive results with TAM, we are aware that agile teams may show resistance to adopt the taxonomy given to the current human dependency for classifying the USs.

As future work, we intend to apply the taxonomy in a more significant number of companies to assess its representativeness in the context of other types of systems such as mobile applications. Moreover, we intend to explore the potential of reuse of other USs related assets such as test cases, non-functional requirements, and others. Furthermore, we plan to expand the database of classified user stories and apply machine learning and natural language processing algorithms to automate the classification procedure.

APPENDIX A TECHNOLOGY ACCEPTANCE MODEL QUESTIONNAIRE

In what follows, the questionnaires applied to operationalize the Technology Acceptance Model in the performed case study are presented.

APPENDIX B TECHNOLOGY ACCEPTANCE MODEL QUESTIONNAIRE

See Table 11.

REFERENCES

[1] R. Capilla, B. Gallina, C. Cetina, and J. Favaro, "Opportunities for software reuse in an uncertain world: From past to emerging trends," *J. Softw., Evol. Process*, vol. 31, no. 8, p. e2217, Aug. 2019.

[2] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, *Manifesto for Agile Software Development*. 2001.

[3] K. Beck and E. Gamma, *Extreme Programming Explained: Embrace Change*. Reading, MA, USA: Addison-Wesley, 2000.

[4] K. Schwaber and M. Beedle, *Agile Software Development With Scrum*, vol. 1. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[5] R. Hoda, N. Salleh, and J. Grundy, "The rise and evolution of agile software development," *IEEE Softw.*, vol. 35, no. 5, pp. 58–63, Sep. 2018.

[6] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents*. Reading, MA, USA: Addison-Wesley, 2003.

[7] X. Wang, L. Zhao, Y. Wang, and J. Sun, "The role of requirements engineering practices in agile development: An empirical study," in *Requirements Engineering*. Berlin, Germany: Springer, 2014, pp. 195–209.

[8] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Comput. Hum. Behav.*, vol. 51, pp. 915–929, Oct. 2015.

[9] Y. Andriyani, R. Hoda, and R. Amor, "Understanding knowledge management in agile software development practice," in *Proc. Int. Conf. Knowl. Sci., Eng. Manage.* Cham, Switzerland: Springer, 2017, pp. 195–207.

[10] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with visual narrator," *Requirements Eng.*, vol. 22, no. 3, pp. 339–358, Sep. 2017.

[11] M. Cohn, *User Stories Applied: For Agile Software Development*. Reading, MA, USA: Addison-Wesley, 2004.

[12] *IEEE Standard for Information Technology–System and Software Life Cycle Processes–Reuse Processes*, IEEE Standard 1517-2010, IEEE Standards Association, Aug. 2010.

[13] T. C. Lethbridge and R. Laganiere, *Object-Oriented Software Engineering*. New York, NY, USA: McGraw-Hill, 2005.

[14] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*. Berlin, Germany: Springer-Verlag, 2005.

[15] C. Palomares, C. Quer, and X. Franch, "Requirements reuse and requirement patterns: A state of the practice survey," *Empirical Softw. Eng.*, vol. 22, no. 6, pp. 2719–2762, Dec. 2017.

[16] X. Franch, C. Quer, S. Renault, C. Guerlain, and C. Palomares, "Constructing and using software requirement patterns," in *Managing Requirements Knowledge*. Berlin, Germany: Springer, 2013, pp. 95–116.

[17] D. Mairiza, D. Zowghi, and N. Nurmuliani, "An investigation into the notion of non-functional requirements," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2010, p. 311.

[18] C. L. Pacheco, I. A. Garcia, J. A. Calvo-Manzano, and M. Arcilla, "A proposed model for reuse of software requirements in requirements catalog," *J. Softw., Evol. Process*, vol. 27, no. 1, pp. 1–21, Jan. 2015.

[19] A. Espinoza and J. Garbajosa, "A study to support agile methods more effectively through traceability," *Innov. Syst. Softw. Eng.*, vol. 7, no. 1, pp. 53–69, Mar. 2011.

[20] R. Elamin and R. Osman, "Towards requirements reuse by implementing traceability in agile development," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2017, pp. 431–436.

[21] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova, "Best practices for automated traceability," *Computer*, vol. 40, no. 6, pp. 27–35, Jun. 2007.

[22] G. Spanoudakis, A. Zisman, E. Pérez-Miñana, and P. Krause, "Rule-based generation of requirements traceability relations," *J. Syst. Softw.*, vol. 72, no. 2, pp. 105–127, Jul. 2004.

[23] J. P. Winkler, J. Grönberg, and A. Vogelsang, "Optimizing for recall in automatic requirements classification: An empirical study," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 40–50.

[24] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 142–152.

[25] R. Prieto-Diaz and P. Freeman, "Classifying software for reusability," *IEEE Softw.*, vol. 4, no. 1, pp. 6–16, Jan. 1987.

[26] R. Prieto-Diaz, "Implementing faceted classification for software reuse," in *Proc. 12th Int. Conf. Softw. Eng.*, 1990, pp. 300–304.

[27] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Inf. Softw. Technol.*, vol. 85, pp. 43–59, May 2017.

[28] F. D. Davis, "A technology acceptance model for empirically testing new end-user information systems: Theory and results," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 1985.

[29] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014. [Online]. Available: https://www.computer.org/education/bodies-of-knowledge/software-engineering

[30] V. Broughton, "The need for a faceted classification as the basis of all methods of information retrieval," *Aslib Proceedings*, vol. 58, nos. 1–2, pp. 49–72, 2006.

[31] D. S. Cruzes and T. Dybå, ''Recommended steps for thematic synthesis in software engineering,'' in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, 2011, pp. 275–284.
[32] D. Šmite, C. Wohlin, Z. Galviņa, and R. Prikladnicki, ''An empirically based terminology and taxonomy for global software engineering,'' *Empirical Softw. Eng.*, vol. 19, no. 1, pp. 105–153, Feb. 2014.
[33] G. R. Wheaton, ''Development of a taxonomy of human performance: A review of classificatory systems relating to tasks and performance,'' Amer. Inst. Res., Pittsburgh, PA, USA, Tech. Rep. 1, 1968.
[34] U. H. Graneheim and B. Lundman, ''Qualitative content analysis in nursing research: Concepts, procedures and measures to achieve trustworthiness,'' *Nurse Educ. Today*, vol. 24, no. 2, pp. 105–112, Feb. 2004.
[35] P. Runeson and M. Höst, ''Guidelines for conducting and reporting case study research in software engineering,'' *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009.
[36] T. Saenphon, ''An analysis of the technology acceptance model in understanding university student's awareness to using Internet of Things,'' in *Proc. Int. Conf. E-Commerce, E-Bus. E-Government (ICEEG)*, vol. 13, 2017, pp. 61–64.
[37] I. C. Swu, S. P. Singh, and Gautam, ''Constraints perceived by broiler farmers in adoption of scientific poultry production practices,'' *Vet. Practitioner*, vol. 13, no. 1, pp. 116–120, 2012.
[38] R. Wieringa, ''Design science methodology: Principles and practice,'' in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. (ICSE)*, vol. 2. New York, NY, USA: Association for Computing Machinery, 2010, p. 493–494.
[39] S. D. Huan, J. C. Yau, E. Tomiak, R. Goel, C. Cripps, S. Z. Gertler, I. A. Prosser, and D. J. Stewart, ''Hydroxyurea did not enhance the clinical response to vinblastine in patients with anthracycline-resistant metastatic breast cancer,'' *Tumori J.*, vol. 82, no. 6, pp. 576–578, Nov. 1996.
[40] P. Sharma and J. Singh, ''Systematic literature review on software effort estimation using machine learning approaches,'' in *Proc. Int. Conf. Next Gener. Comput. Inf. Syst. (ICNGCIS)*, Dec. 2017, pp. 43–47.
[41] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer-Verlag, 2012.

EDNALDO DILORENZO received the M.Sc. degree in computer science from the Federal University of Pernambuco, Brazil, in 2012. He is currently pursuing the Ph.D. degree with the Federal University of Campina Grande. He is also a Professor with the Federal Institute for Education, Science, and Technology of Paraíba (IFPB), and a Researcher with the Intelligent Software Engineering Group (ISE/VIRTUS). His research interests are in the application of intelligent techniques to software engineering problems.

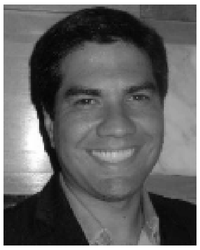
EMANUEL DANTAS received the M.Sc. degree in computer science from the Federal University of Fortaleza, Ceará, Brazil, in 2014. He is currently pursuing the Ph.D. degree with the Federal University of Campina Grande, Paraíba, Brazil. He has been a Professor with the Federal Institute of Paraiba, since 2015. He is also a Researcher with the Intelligent Software Engineering Research Group, VIRTUS, which is the Research, Development, and Innovation Center in Information Technology. His current research interest is in artificial intelligence applied to software engineering to solve complex problems. In the software engineering field, the main topics of interest are software project management, agile software development, effort estimation, risk management, and others.

MIRKO PERKUSICH received the Ph.D. degree in computer science. He is currently a Research Manager with VIRTUS Innovation Center, leading the Intelligent Software Engineering Research Group. His current research interests are in applying intelligent techniques, including recommender systems, to solve complex software engineering problems, with over 50 published articles.

FELIPE RAMOS received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Campina Grande, Paraiba, Brazil, in 2012 and 2019, respectively. He is currently a Professor with the Federal Institute of Paraiba. He is a member of the Intelligent Software Engineering Research Group, VIRTUS, which is the Research, Development, and Innovation Center in Information Technology. His current research interests are in artificial intelligence applied to software engineering to solve complex problems. His main topics of interest are in agile software development and requirement engineering focused on supporting the elicitation of non-functional requirements on scrum-based projects.

ALEXANDRE COSTA received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Campina Grande, Paraiba, Brazil, in 2014 and 2019, respectively. He has been a Professor with the Federal Institute of Paraiba, since 2020. He is currently a Researcher with the Intelligent Software Engineering Research Group, VIRTUS, which is the Research, Development, and Innovation Center in Information Technology. His current research interests are in artificial intelligence applied to software engineering to solve complex problems. In the software engineering field, his main topics of interest are software project management, agile software development, resource allocation focused on team formation for software development, and others.

DANYLLO ALBUQUERQUE received the M.Sc. degree in informatics from the Federal University of Paraiba, Brazil, in 2013. He is currently pursuing the Ph.D. degree in computer science with the Federal University of Campina Grande. He has been a Professor with the Federal Institute for Education, Science, and Technology of Paraiba (IFPB), since 2020. He is also a Systems Analyst with the Federal University of Campina Grande. He is a member of the Intelligent Software Engineering Research Group, VIRTUS, which is a Research, Development, and Innovation Center in Information Technology. His current research interests are in software quality, technical debt, software architecture, artificial intelligence, as well as intelligent techniques applied to solve complex software engineering problems.



HYGGO ALMEIDA received the Ph.D. degree in electrical engineering and the M.Sc. degree in computer science from the Federal University of Campina Grande, in 2007 and 2004, respectively. He has been a Professor with the Computer and Systems Department, Federal University of Campina Grande (UFCG), since 2006. He is currently the Head of the Intelligent Software Engineering Group, and the Founder and the Director of Operations with the VIRTUS Innovation Center (VIRTUS/UFCG). He is also a Researcher with the Embedded and Pervasive Computing Laboratory (Embedded/UFCG). He is also the Executive Director of the EMBRAPII Unit, CEEI-UFCG, with more than 150 RD&I projects developed in cooperation with industrial companies within the area of information, communication, and automation technologies. He has more than 15 years of teaching experience in the university as well as training courses for industry in the context of software engineering. He has more than 200 articles published, 37 master thesis, and 13 doctoral dissertations advised. His current research interest is applying intelligent techniques to solve complex software engineering problems.

ANGELO PERKUSICH (Member, IEEE) received the master's and Ph.D. degrees in electrical engineering from the Federal University of Paraíba, in 1987 and 1994, respectively. He was a Visiting Researcher with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA, from 1992 to 1993, and has developed research activities on software engineering and formal methods. He has been a Professor with the Electrical Engineering Department (DEE), Federal University of Campina Grande (UFCG), since 2002. He is currently the Principal Investigator of research projects financed by public institutions, such as FINEP (Brazilian Agency for Research and Studies) and CNPq (Brazilian National Research Council), as well as private companies. He is also the Founder and the Director of the VIRTUS Innovation Center and Embedded and Pervasive Computing Laboratory. His research projects focus on formal methods, embedded systems, mobile pervasive and ubiquitous computing, and software engineering. He has more than 30 years of teaching experience in the university as well as training courses for industry in the context of software for real-time systems, software engineering, embedded systems, computer networks, and formal methods. His main research areas are embedded systems, software engineering, mobile pervasive computing, and formal methods, with more than 300 articles published, 80 master thesis, and 21 doctoral dissertations advised.