International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 11 | Nov - 2021 ISSN: 2582-3930

Automating User Stories Classification Based on Machine Learning Algorithm

Mitra Malekzad[1], Dr. K. Madhavi[2]

[1]*M-Tech Scholar, Department of Computer Science and Engineering, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India*

[2]*HOD and Associate Professor, Department of Computer Science and Engineering, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India*

Abstract - 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). Therefore, this paper proposes 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. 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. The proposed taxonomy for USs is composed of two sub-facets, namely, module and operation, which have, respectively, three and 18 terms

Key Words: Agile software development, Reuse-Driven Software Engineering, software reuse, user stories, information retrieval.

1. 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]. 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. In ASD the main asset that stores product knowledge is the User Story (US). The US consists of a semi-structured notation to specify requirements. The main goal of the US is to facilitate product-related discussions between the stakeholders. 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, 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. By reusing requirements, there is the potential to have a baseline to address some interesting questions such as: Was it efficient? How can we do better?"
if we analyze companies that work on products of the same type, 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?

Table-1: US distribution over module 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

2. LITERATURE SURVEY

Opportunities for software reuse in an uncertain world: From past to emerging trends

Much has been investigated about software reuse since the software crisis. The development of software reuse methods, implementation techniques, and cost models has resulted in a significant amount of research over years. Nevertheless, the increasing adoption of reuse techniques, many of them subsumed under higher level software engineering processes, and advanced programming techniques that ease the way to reuse software assets, have hidden somehow in the recent year's new research trends on the practice of reuse and caused the disappearance of several reuse conferences. Also, new forms of reuse like open data and feature models have brought new opportunities for reuse beyond the traditional software components. From past to present, we summarize in this research the recent history of software reuse, and we report new research areas and forms of reuse according to current needs in industry and application domains, as well as promising research trends for the upcoming years.

The results of our practitioner survey have made it abundantly clear that Rubén Prieto-Diaz was right in his prediction that "software reuse will cease to exist": In the 26 years since the original survey was performed, practitioners have learned to integrate software reuse so completely into their development processes that it is no longer even a matter of discussion. But were the predictions right that software reuse would cease to exist as an evolving, research-worthy discipline? Both the vibrant industrial sector reuse communities and the research trends described in this paper indicate that the answer is emphatically No. Our ambition continues to drive us to harness emerging technologies to pursue reuse research in new contexts, with new artifacts, and for new, even more powerful search, retrieval, and development processes. In conclusion, we cite one final prediction, made over 30 years ago by Fred Brooksm,78 that reuse would remain the primary means of confronting the essential complexity of developing software. The implications of this are clear: No matter where our ambition takes us in the future, uncertain world, reuse will always be with us, and there will always be new opportunities for software reuse.

The role of requirements engineering practices in agile development: An empirical study

Requirements Engineering (RE) plays a fundamental role in all sorts of software development processes. Recently, agile software development has been growing in popularity. However, in contrast to the extensive research of RE in traditional software development, the role of RE in agile development has not yet been studied in depth. In this paper, we present a survey with three research questions to explore the treatment of RE in the practical agile development by investigating eight agile groups from four software development organizations. To answer the three research questions, we targeted at 108 participants with rich agile experiences and designed a questionnaire to collect their answers. Our survey shows that agile RE practices play a crucial role in agile development and they are an important prerequisite for projects' success though many agile methods advocate coding without waiting for formal requirements and design specifications.

Our study reveals that RE practices play a crucial role in agile development. Although many agile methods advocate coding without waiting for formal requirements and design specifications, RE is still an important prerequisite for the success of projects. The value of the work presented in this paper is the identification of a set of following findings about RE practices for agile practitioners. Lack of concern on RE in practice. Many project delays were caused by insufficient communication with customers or shortage of RE resources. The role of agile RE practices has been acknowledged by most agile practitioners, nevertheless, they didn't make an even resource allocation due to the project cost. Therefore, it is recommended that agile groups should carefully leverage the resources assigned to RE and the costs. Lack of concern on NFRs. Agile practitioners should pay more attention to NFRs at the beginning of the project, rather than leave them to design or coding. We suggest them to adopt some methods such as NFR framework for NFRs analysis. Preference for agile RE practices. In agile RE practices, the participants identified that interviewing and user story are the most important requirements elicitation practices whereas user story and use case are the most widely used requirements representation practices. In order to rapidly capture requirement changes from customers, it is important to intensively communicate with customers. Although agile RE differs from traditional RE in that it takes an iterative discovery approach, many traditional RE practices are still applicable in agile development such as use case, modeling and prototyping and so on.

3.PROPOSED SYSTEM

This paper aims to propose 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. We constructed 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. The proposed taxonomy for USs is composed of two sub-facets, namely, module and operation, which have, respectively, three and 18 terms.

Initially, we filtered the product backlog items to be analyzed, by removing the ones that were not related to functional requirements for instance, we removed product backlog items related to technical debt (e.g., refactor a component) or nonfunctional requirements (e.g., add support to internationalization and localization). 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. In this work, we will use text normalization, feature extraction techniques and machine learning algorithms to create US classification model, using software repository of Cassandra and Eclipse.

Table 2: 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

4.Method and IMPLEMENTATION

we define 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. For the structure of the taxonomy, it is well-known that facet-based classification is adequate to organize reusable software development assets 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''. The procedure to classify the user stories is qualitative and based on thematic analysis. Finally, the basis (i.e., sources of information) to define the taxonomy are product backlogs from real-world projects.

In this work, we perform classification of USs by Bag of Words which is the best text vectorization technique and Random Forest machine learning model which has the best performance in the task of classifying requirements.

Algorithms

Algorithm 1 Similarity Between User Stories ui and uj,

1: *procedure* SIMILARITY(ui,uj,M)

2: **if** ui and uj are mapped to at least one common ck *then*

3: *return* TRUE F ui and uj are similar.

4: *else*

5: *return* FALSE F ui and uj are not similar.

Classification Method

We used four phases (steps) to perform the US classification

1. Normalization
2. Vectorization
3. Classification.
4. Evaluation.

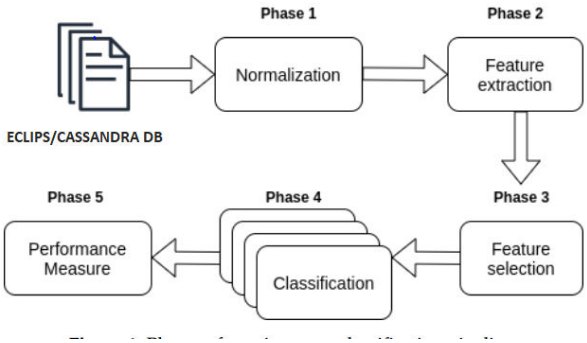

Fig-1: User Story Classification method

System Architecture

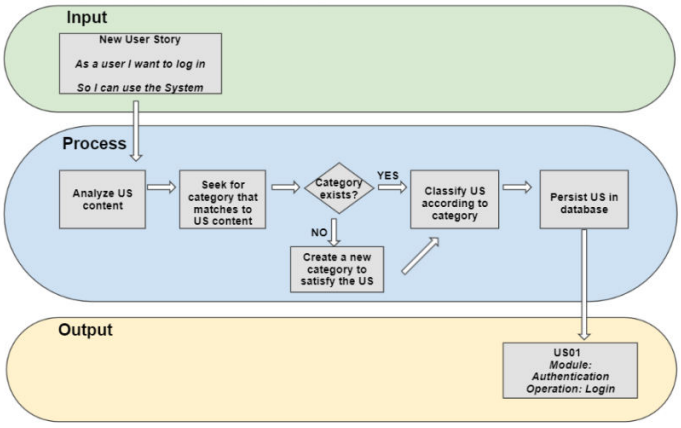

Fig-2: Process to classify USs based on the proposed taxonomy

5. EXPRIMENTAL RESULT

The proposed taxonomy and methodology has been discussed. The aim of this taxonomy is to enable reuse of user stories in Agile methodology to ease and speed up the process of software development. The below figures show the process of identifying, and classifying user stories in order to save them in a database and reuse them for the further software projects. As shown in the figure 3, we performed the conversion of the US into numerical vectors, each of these vectors will combined with the classification algorithms.

Generated Taxonomy Features Vector

	able	abortcompilation	abortcompilationunit	abortexception	...	zip	zipentry	zipexception	zipf
0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
..	...	...	...	...	...	...	...	...	
208	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
209	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
210	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
211	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
212	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

[213 rows x 5395 columns]

Fig-3: User Story conversion into numerical vectors

The below fig-4 shows random forest machine learning algorithm which got 93% correct prediction accuracy out of 100.

Machine Learning Random Forest Train & Test dataset split
80% user stories records used to train ML and 20% records used to test ML prediction accuracy
Total user stories found in dataset is : 213

Machine Learnig Training Records Length : 170

Machine Learnig Testing Records Length : 43

Random Forest Prediction Accuracy on test records : 90.69767441860465

Fig-4: Result of random forest machine learning algorithm

We are required to enter a user story to classify/predict matching user story as shown in fig-5.

Fig-5: Result of adding new user stories

Finally, the taxonomy finds out the program name from dataset folder which contains matching user story.

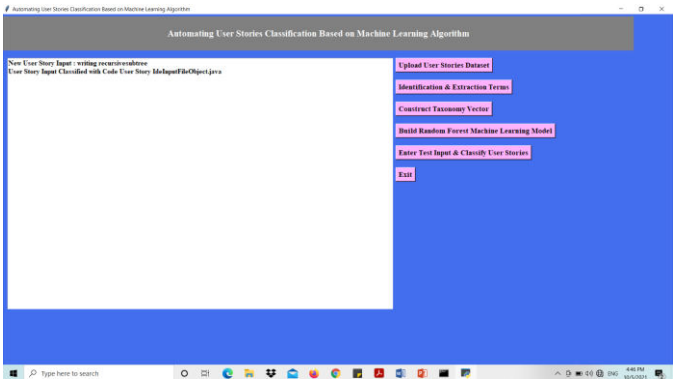

Fig-6: Result of User story classification

Table-3: Sample of classified task of taxonomy

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

6. CONCLUSIONS

This article proposed a taxonomy for user stories and evaluated its potential to promote reuse-driven software engineering for agile software development. 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. We constructed 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).

ACKNOWLEDGEMENT
The authors would like to express their gratitude to the contributors of the ECLIPSE and CASSANDRA repository's publicly accessible frameworks, libraries, and sketching datasets, which aided training, testing, and validation.

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] 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.

[3] 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.

[4] 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

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

[9] . 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.

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