Enhancing Software Development Efficiency Through User Stories Reuse from Application Descriptions

Indra Kharisma Raharjana
Center for Information Systems Engineering
University of Airlangga, Indonesia
indra.kharisma@fst.unair.ac.id

Avril Hermawan
Faculty of Science and Technology
University of Airlangga, Indonesia
avrilaudihermawan@gmail.com

Badrus Zaman
Faculty of Science and Technology
University of Airlangga, Indonesia
badruszaman@fst.unair.ac.id

Shukor Sanim Mohd Fauzi
Faculty of Computer and Mathematical Sciences
University of Teknology MARA, Malaysia
shukorsanim@uitm.edu.my

Abstract: *Software development projects frequently encounter cost and time efficiency challenges, rendering them susceptible to failure. An imprecise and incomplete process of eliciting user software requirements can further escalate software development's time, cost, and effort. Consequently, the reuse of user stories based on application descriptions for software requirement elicitation is proposed to mitigate these issues and enhance cost efficiency. This research aims to evaluate the reuse of user stories within the software elicitation process by extracting software features from initial application descriptions. The goal is to determine whether this approach can effectively reduce the time and cost associated with software development. This study employs Natural Language Processing (NLP) techniques to identify Part-Of-Speech (POS) tag patterns to extract software features from application descriptions. A similarity measurement is conducted between user stories and the extracted software features. User stories are then filtered based on a similarity threshold to ensure high relevance to the extracted software features. The proposed method was tested using three distinct application descriptions alongside 22 user story datasets provided to nine respondents. The analysis indicates that the reuse of user stories based on feature extraction from application descriptions, determined through Cosine similarity with an 80% threshold, significantly enhances software development efficiency in terms of time and cost. The method demonstrated an average precision of 84%, recall of 93%, and F1-score of 86% across the three test projects, confirming its alignment with system design requirements. The findings of this research validate the effectiveness of the Cosine similarity method in identifying reusable user stories from application descriptions. This approach is a viable solution for reusing user stories in the software requirement elicitation process, thereby improving efficiency in software development projects.*

Keywords: *Application description, requirements reuse, requirements elicitation, process innovation, user story.*

Received July 31, 2024; accepted August 7, 2025
https://doi.org/10.34028/iajit/23/1/4

1. Introduction

Requirements elicitation is the process of finding, uncovering, obtaining, and describing requirements for the software development process [40]. Requirement elicitation is the initial process of the software development process [25]. In practice, requirements elicitation is a process that has a significant role in software development because if the requirements in requirements elicitation are incomplete, it can increase costs and efforts in software development [29]. Extraction of requirements obtained from various sources of unstructured documents increases software development efficiency [30]. Reusing practices allows for reducing costs and efforts in the requirements elicitation [27]. Requirements reuse is reclaiming software requirements developed and used in earlier projects [15].

Requirement reuse is an effective method for increasing efficiency in the software development process, reducing software development and maintenance costs, producing higher-quality software products, improving software and system dependencies, and facilitating the transfer of team members, tools, and methods between projects [11]. In general, requirements reuse is extracted from the Software Requirements Specification (SRS), but not everyone can do SRS extraction for requirements reuse because not everyone has access to software artifacts [2, 20]. Apart from being obtained from SRS, reuse requirements can be extracted from user reviews and application descriptions. User reviews are ratings written by users; in addition to ratings, some users suggest new features that users want.

Meanwhile, the developer writes the application description containing complete information about the application name, features, and benefits. It aims to help users quickly find the application they need [14]. In this study, the application description is the source document that will be extracted to obtain software features because it can be written at the beginning of the software development process and accessed by everyone, in contrast to SRS, which is not accessible to everyone and user reviews which can only be obtained when the software has been released, used, and reviewed by users.

Several methods can be used in software development. One is agile development software, a short-term software development that requires developers to adapt quickly to various changes [16, 26, 39]. In agile, user stories collected during the requirements elicitation are the most widely used artifacts to obtain simple descriptions from the user's perspective [35]. A user story is a short, structured sentence requesting user needs or functions [6]. User story consists of 3 aspects: who/who asks for the needs or functions, what or what needs or stakeholders request functions, and why/why stakeholders want these functions [18, 36]. User stories are widely used during requirements elicitation because user stories play a role in explaining the needs and desires of stakeholders or personas [21]. User stories can be boundary objects in transferring and creating new knowledge [7].

User stories are widely used during the requirements elicitation in agile software development [10, 34]. User stories that have been collected are generally not collected again by software developers, which opens up opportunities for developers to reuse user stories collected from several previous projects for future projects [7]. Reusing user stories based on application descriptions for software elicitation needs aims to reduce costs and efforts in software development [7, 24]. The practice of reuse can reduce costs and efforts in the requirements elicitation process [27]. Customer satisfaction results from cost and time efficiency [4]. The benchmarks for the success of agile software development are time, cost, effort, and customer satisfaction [1, 33]. The purpose of this study is to reuse user stories for the elicitation of software based on extracting software features from the initial application description. This research produces an output list of user stories that can be reused; the potential is obtained from the similarity value of user stories with the extraction of software features from the application description.

2. Related Works

Software developers usually write app descriptions and describe their features on the app store, and users comment about the features in the app reviews section [37, 38]. Extracting and matching functionality from app descriptions is critical to benefiting from the App Store. In the study of Johann *et al.* [14], propose a new Approach for Extracting Software Features (SAFE) from application descriptions and user reviews and matching them. This research uses 18 parts of speech written manually and five sentence patterns that often refer to software features. This study will extract application descriptions to obtain software features using part-of-speech. Software features that have been extracted will be searched for compatibility with the processed user story. There are several techniques in requirements elicitation to increase understanding of domain knowledge, such as user interviews, questionnaires, document analysis, and brainstorming. Most of these techniques require in-depth stakeholder engagement. In practice, stakeholders generally have limited time and stakeholder so that not all software projects can use this technique [31]. Software development is agile, and user stories are used to capture and write functional requirements. A user story is an appropriate and easy-to-understand format for writing the results of a requirement. Research by Raharjana *et al.* [30] aims to propose a model for extracting user stories from online news to increase understanding of domain knowledge.

User stories are a fundamental component of agile software development, which captures functional requirements from the end-user's perspective. In this context, reusable user stories refer to user stories that can be applied across different projects or domains with minimal modification [17]. The reusability of user stories helps improve efficiency and reduce excessive effort in software development. In this study, user stories are used to reuse the requirements. This is because user stories are widely used during requirement elicitation in agile software development so that they can be reused to increase cost efficiency and software development efforts.

High-quality user stories are typically characterized by the INVEST criteria: Independent, Negotiable, Valuable, Estimable, Small, and Testable [3]. These attributes ensure that user stories are well-structured and manageable in agile development. While our study focuses on automation rather than manual refinement, the INVEST principles provide a valuable benchmark for assessing the quality of extracted user stories. Feature sets are essential assets for reuse in software product line methodologies. In requirements reuse, the extraction of software features from a SRS can only be performed by practitioners who have access to these software artifacts. Due to organizational privacy, the SRS is always kept confidential and is not easily used by the public. As an alternative, researchers use publicly available software descriptions (such as product manuals and online) to identify potential software functions to initiate the reuse of requirements. Bakar *et al.* [2] research aims to propose a semi-automated method, called Feature Extraction for Natural Language Reuse (FENL), to extract phrases that can represent software functions from software reviews without using SRS as a method to initiate the technique. This study will use application descriptions as document sources to obtain software features. Software features will be used to reuse the requirements. The extraction process from the application description requires a pre-processing as described in the research by Bakar *et al*. [2].

3. Method

The research step begins with extracting software features in the application description by pre-processing the application description by tokenizing sentences and cleaning data. The next step is labeling each word with Part-Of-Speech (POS) tagging. After being marked on each sentence, an extraction process takes the word fragments labeled according to the needs and then collects them as a list using Chungking POS. Then, do Case Folding, which changes all the letters on the list to lowercase. After case folding, changing becomes the base word for each word on the list using WordNet Lemmatization. After being converted into basic words, words without meaning are removed, usually called stop words. The next step of this research is to extract the user story dataset, starting with separating the sentences in the user story. Then, noise such as punctuation and numbers are removed, followed by case folding, lemmatization, and stop word removal.

The last step of this research is filtering the user story related to the application description and then calculating the similarity between software feature extraction and user story extraction. User stories that have a similarity exceeding the threshold will be included in the list of user stories that can be reused. Furthermore, an evaluation will be conducted to assess the information's accuracy. The architecture of the proposed method can be seen in Figure 1.

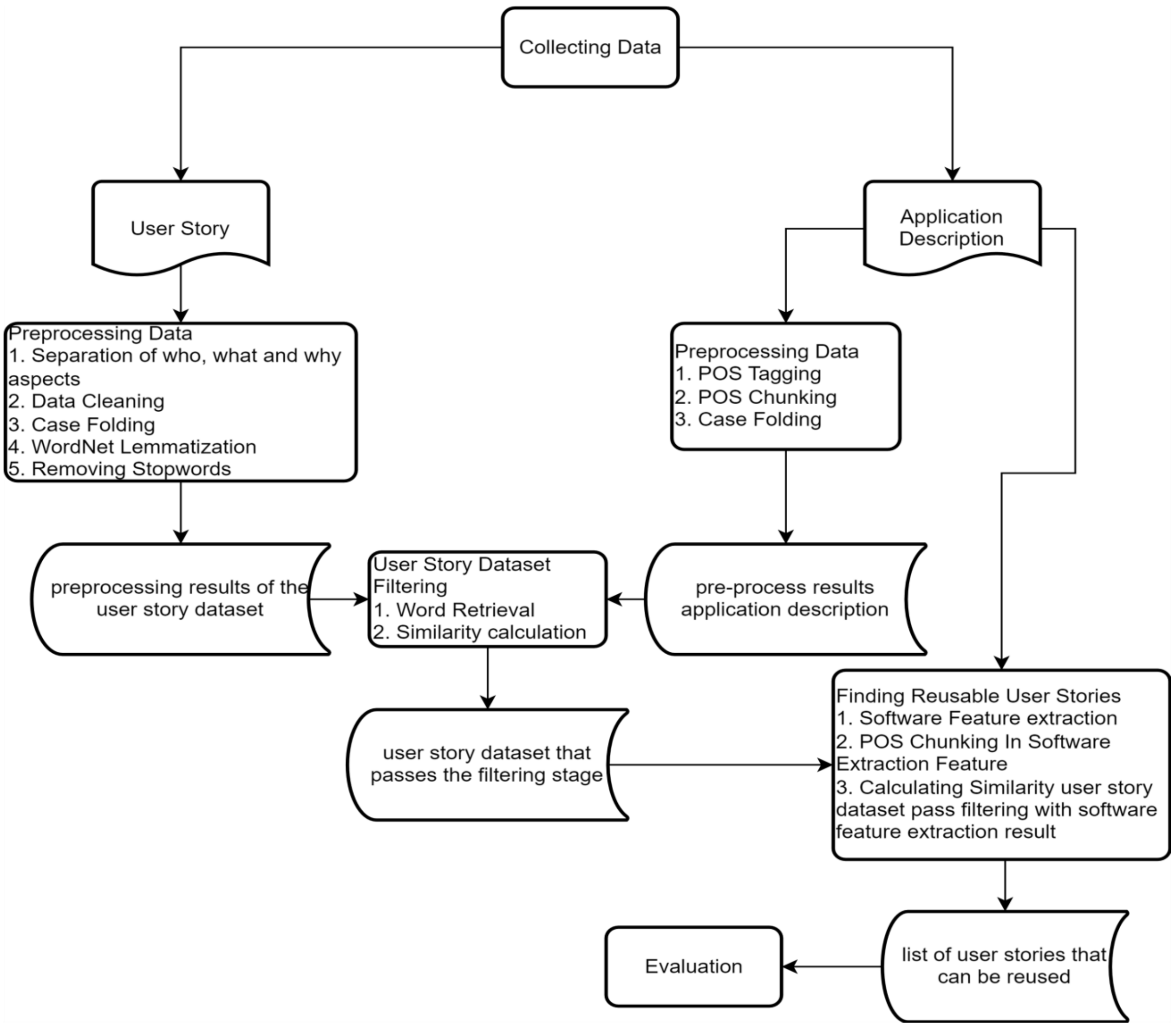
Collecting Data
User Story
Application Description
Preprocessing Data
1. Separation of who, what and why aspects
2. Data Cleaning
3. Case Folding
4. WordNet Lemmatization
5. Removing Stopwords
Preprocessing Data
1. POS Tagging
2. POS Chunking
3. Case Folding
preprocessing results of the user story dataset
User Story Dataset Filtering
1. Word Retrieval
2. Similarity calculation
pre-process results application description
user story dataset that passes the filtering stage
Finding Reusable User Stories
1. Software Feature extraction
2. POS Chunking In Software Extraction Feature
3. Calculating Similarity user story dataset pass filtering with software feature extraction result
Evaluation
list of user stories that can be reused

Figure 1. Architecture of the proposed method.

3.1. Collecting Data

This study requires three types of data: user story datasets, application descriptions, and expert evaluations. The user story dataset is the source of reusable artifacts in the software elicitation process. Application descriptions are extracted to obtain software features, while expert data is used for evaluation and comparison.

User stories typically comprise three aspects: who, what, and why [18]. Some user stories may not explicitly include the why aspect. To extract meaningful information, user stories are processed by separating these aspects: identifying the user (who) requested features (what) and the rationale or benefit of the feature (why). The dataset [31] includes over 1,500 user stories from 22 applications covering various domains. Extraction steps involve tokenization, data cleaning, case folding, WordNet lemmatization, and stopword removal. Application descriptions contain key information about an application's name, features, and benefits, written by developers to attract users [14]. A well-written application description is concise, informative, and highlights key features effectively [13]. This study extracts these descriptions to derive software features, which are then used to filter user stories. Word extraction uses POS tagging, chunking, and case folding techniques.

3.2. Pre-Processing Data

The next stage involves pre-processing user story and application description data to ensure consistency and improve processing efficiency. Pre-processing includes tokenization, case folding (converting text to lowercase), lemmatization (reducing words to their base forms), and stopword removal (eliminating common but non-essential words). These steps refine the data for further analysis in the research workflow.

The data pre-processing stage ensures that user stories and application descriptions are structured for further analysis. This process includes aspect separation, data cleaning, POS tagging, chunking, case folding, lemmatization, and stopword removal. User stories are segmented into three key aspects: who (the stakeholder or end-user requesting the functionality), what (the requested feature), and why (the reason or benefit) [35]. The separation is performed using delimiters such as commas (,) and the word "so". Next, data cleaning removes unnecessary elements such as numbers, punctuation, and symbols to reduce noise. POS tagging is then applied to label words based on their grammatical roles, facilitating further text processing. Chunking is performed to group related words into meaningful phrases using predefined rules.

Case folding standardizes text by converting all characters to lowercase, ensuring consistency in text analysis. Lemmatization further refines the data by reducing words to their base forms, helping to improve the accuracy of similarity calculations between extracted user stories and software features. Finally, stopword removal eliminates common but non-essential words such as "as," "is," and "all," streamlining the dataset for more effective processing.

3.3. User Story Dataset Filtering

In this stage, the user story dataset is filtered to enhance system efficiency by reducing errors, improving processing speed, and increasing accuracy. The filtering process evaluates the similarity between extracted user stories and words from application descriptions. Several methods are employed to select relevant words from application descriptions. The first method identifies frequently occurring words, while the second focuses on extracting nouns. Both approaches utilize TF-IDF for word weighting to prioritize important terms. Additionally, the YAKE tool is used to extract automatic keywords. Cosine similarity is applied to measure the closeness between extracted words and the user story dataset to determine relevance. The user story is considered relevant for further processing if the similarity score exceeds a predefined threshold.

3.4. Finding Reusable User Stories

Each story is compared with the software feature extraction from application descriptions to identify reusable user stories. The extraction process involves separating application descriptions into individual sentences. While this process shares similarities with pre-processing, the chunking step follows distinct rules tailored for feature extraction. Chunking uses part-of-speech patterns and regular expressions to group relevant words and phrases. These extracted phrases are then analyzed for similarity with user stories using Cosine similarity and Jaccard similarity methods. If a user story's similarity score exceeds a predetermined threshold, it is considered relevant to the extracted software features. The final output is a structured table displaying user stories that can be reused based on similarity calculations. This table is then evaluated to assess the accuracy and effectiveness of the proposed method.

3.5. Evaluation

Evaluation is carried out to assess whether the system created can provide accurate information about the list of user stories that match the application description. The first step in evaluating this system is to determine an evaluation case. The determination of the topic description of the application is required to follow the user story dataset. So, a user story in the dataset has similarities and can be reused. Next, the expert manually marks the user stories that are considered to have similarities with the application description. The last step is to perform precision and recall tests on manual results from experts with results from the system. Experts here have been involved in software development projects as programmers or system analysts. Evaluation is carried out with a minimum of 3 experts.

Precision measures the accuracy between the list of reused user stories written by experts and the list provided by the system or the system's accuracy in classifying data. Calculating the *precision* value can be done by Equation (1).

$$Precision = \frac{|\{relevant\ document\} \cap \{Retrieved\ Document\}|}{|\{retrieved\ documents\}|} \quad (1)$$

Recall compares the number of lists of reused user stories correctly by the system in a class against all actual data. The *recall* calculation can be seen in Equation (2).

$$Recall = \frac{|\{relevant\ document\} \cap \{Retrieved\ Document\}|}{|\{relevant\ Documents\}|} \quad (2)$$

- *Relevant document*: a list of reusable user stories written by experts.
- *Retrieved document*: a list of reusable user stories generated by the system.

4. Result

In this section, an explanation will be given regarding the research results carried out while conducting the research. The explanation given is an elaboration of the existing problem formulation.

4.1. Data Collection

The user story datasets were sourced from Mendeley, where they were publicly uploaded in 2018 [5]. There are 22 applications in the user story dataset with various topics. In this study, it is necessary to understand the subject of each application to determine a description of similar applications. The detail of the user story dataset can be seen in Table 1.

Table 1. User story datasets topics.

Project name	Project topic
Federalspending	Monitor state spending.
Loudoun	Loudoun city planning portal for residents and visitors to the city.
Recycling	Help find recycling places.
Openspending	Monitor and study state finances.
Frictionless	Describe validate (process) data.
Scrumalliance	Provide learning about scrum and agile.
Nsf	Research funding and awards.
Camperplus	Camping service provider.
Planningpoker	Game for agile planning.
Datahub	Data Collection.
Mis	Duke university information management and repository.
Cask	Software development portal.
Neurohub	Data storage and intermediary Research collaboration.
Alfred	Oldster assistance.
Badcamp	Event management and registration.
Rdadmp	Data management planning.
Archivesspace	Archiving.
Unibath	University of Bath Repository.
Duraspace	Repository for digital content.
Racdam	Archiving.
Culrepo	Content management for universities.
Zooniverse	Portal to provide support for research.

4.1.1. Application Description

A description of the application is obtained by looking for a software project similar to the user story. The software project topics used in this research are recycling, digital assistants for parents, and digital archiving. The search is done using a search engine from Google by entering the keywords “software project” or “project idea,” followed by the topic to be searched in English. Google will show a collection of pages that have software projects with relevant topics. In this study, application descriptions were sorted manually by looking at the relevance of the software project on the page. In this study, the description page of the software project will be opened automatically using the software package “trafilatura.”

4.1.2. Expert Data

The experts here have been involved in software development projects as programmers or system analysts, and nine experts were needed in this study. The first stage of the data collection process is when the author explains the expert procedures for online data annotation via video communication (Google Meet). The video communication explains that the expert will be given three application descriptions and 22 user stories containing 1677 user stories. Then, the expert will be given a brief description of each user story. Then, the expert will annotate the number of 0 (which means it is irrelevant to the application description) and 1 (relevant to the description) from 1677 user stories.

4.2. Data Extraction

The second stage in this research is the pre-processing of collected data.

4.2.1. User Story Extraction

This user story extraction stage includes tokenization, case folding, data cleaning, lemmatization, and stopword removal. Separation of user story sentences into two or three parts depending on the availability of the number of aspects in the user story. User stories are separated based on their aspects, namely who (who) wants this functionality, what functional (what) stakeholders or end-users wish to from the system, and why (why) stakeholders or end-users need this function. The separation is done by making a regular expression with a pattern set according to research needs. The aspects used in this study are the who and why aspects. Case folding is changing all letters in the user story dataset to lowercase. The step to do case folding is to call the lower () function in each iteration of the requirements list document. Data cleaning is done because the user story dataset contains many disturbing characters for the next stage. These characters include punctuation at the end of words and other irrelevant characters. Data cleaning is done by calling the remove punctuation () function. Lemmatization is the process of grouping words in different forms to be analyzed as one item. In this process, the additional affixes of the word will be removed and converted into basic word forms. Lemmatization is carried out to reduce the vocabulary in the user story dataset and application description, which is expected to increase the similarity value of the user story dataset extraction with software feature extraction. Stopwords are the most common words in the language, for example, like, “as,” “a,” “I,” “is,” “all.” Therefore, these words have no meaning and are removed from the text of the user story dataset and application description.

4.2.2. Application Description Word Extraction

Word extraction is intended to retrieve words that are considered necessary from an application description. To get accurate results, it is necessary to experiment using three-word extraction methods to get the most accurate and efficient results. This study uses three trial methods: TF-IDF, TF-IDF for nouns in the user story dataset, and tools from YAKE, which can extract words automatically.

- **TF-IDF**: word extraction is done by looking for the word that appears most often and weighing the top 20 words using TF-IDF.
- **TF-IDF and Nouns phrases**: word extraction intended to retrieve nouns in the application description. The methods used in noun extraction are case folding for changing the shape of each letter in the same form and POS tagging Natural Language Processing (NLP) to assign word labels to sentences automatically. POS tagging uses the pos_tag function in the Natural Language Toolkit (NLTK) library for each review sentence iteration, and POS Chunking to group words or phrases in a sentence into a new group called 'chunks' using adjusted rules. In this stage, the rules used to retrieve nouns are words with tags Noun (NN), Proper Noun (NNP), NNs, and NNPs.
- **YAKE!**: Yake is a new feature-based system for multi-language keyword extraction that supports text of any size, domain, or language.

4.3. Filtering User Story Dataset

The third stage in this research is filtering or selecting the user story that will be used for further processing. Filtering is done to reduce system errors, speed up processing time, and increase the system's accuracy to make the system much more efficient. Filtering is done by looking for the closeness between user story extraction and word extraction in the application description.

4.3.1. Finding the Similarity of Word Extraction with User Story

After the extraction process is carried out, the three methods at the application description extraction stage, namely word extraction using TF-IDF, noun extraction using TF-IDF, and YAKE, will be calculated for proximity using the Cosine similarity method to calculate the similarity between the application description and the user story dataset that has been extracted. Then, sort the user story dataset based on the magnitude of the similarity value with the application description displayed in the data frame, as shown in Table 2. In Table 2, the user story recycle dataset has the most prominent similarity value, with a score of 0.16, so the dataset can be reused compared to other datasets with lower scores.

Table 2. Result of similarity score of application descriptions with user story.

#	App description	User story	Similarity score
0	daur	recycling	0.161061
1	daur	nsf	0.134181
2	daur	culrepo	0.126633
3	daur	neurohub	0.118156
4	daur	rdadmp	0.112035
5	daur	mis	0.099074
6	daur	unibath	0.097821
7	daur	alfred	0.092581
8	daur	racdam	0.08904
9	daur	openspending	0.089037

4.3.2. Testing the Dataset Filtering Method

To determine the number of user story datasets used and which method is most appropriate for this study, an experiment was carried out by comparing the results from the system with the results of data retrieval from the evaluator. The first step is to combine the data obtained from nine evaluators. Merging is done by taking the user story dataset labeled 1 (considered relevant) by more than 50% of the nine evaluators. A collection of combined user stories will be used as a reference in this study's appropriate method trials and evaluation process. The retrieval of datasets in this test uses 2, 5, and 10 datasets to make a significant difference in test results.

4.3.3. Calculating the Accuracy and Success of the Method

To find out the correct method for this study and the number of datasets taken for the filtration process, precision-recall and F1-scores were manually calculated between the evaluator's and the system's results. The result of this calculation can be seen in Table 3.

Table 3. Results of similarity score of application descriptions with user story.

Project	Methods	2 Dataset			5 Dataset			10 Dataset		
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Recycle	TF-IDF + noun phrase	0,5	1	0,667	0,2	1	0,333	0,1	1	0,182
	YAKE	0,5	1	0,667	0,2	1	0,333	0,1	1	0,182
	TF-IDF	0	0	0	0	0	0	0,1	1	0,182
Archive	TF-IDF + noun phrase	1	0,333	0,5	0,8	0,66	0,727	0,4	0,66	0,55
	YAKE	0	0	0	0,2	0,167	0,182	0,2	0,333	0,25
	TF-IDF	0	0	0	0,2	0,167	0,182	0,2	0,333	0,25
Alfred	TF-IDF + noun phrase	0,5	1	0,667	0,2	1	0,333	0,1	1	0,182
	YAKE	0,5	1	0,667	0,2	1	0,333	0,1	1	0,182
	TF-IDF	0	0	0	0	0	0	0	0	0

The recycle project uses the TF-IDF method for nouns, according to Table 2. Using two datasets produces a precision value of 0.5, a recall of 1, and an F1-score of 0.6667. This figure is obtained by performing manual calculations by calculating the results according to the evaluator with the output generated by the system. Based on these three projects, the TF-IDF weighting method based on noun extraction produces the most significant value by taking two datasets on the recycle and Alfred projects. Taking five datasets has the most substantial value in the project archive. The YAKE method produces the most significant value based on noun extraction by taking two recycle and Alfred projects datasets.

Furthermore, taking ten datasets has the most substantial value in the project archive. The TF-IDF weighting method based on noun extraction produces the most significant value by taking ten datasets on the Recycle and Archive projects. This method did not identify datasets related to the application description in the Alfred project. There is a difference in the number of datasets used because, in the Recycle and Alfred projects, the number of datasets considered relevant by the evaluator is only one. Six datasets for the archive dataset project are regarded as appropriate.

It can be concluded from the experimental results above that the TF-IDF method with noun extraction is the most appropriate compared to the other two methods. The results of this filtration process are considerable time efficiency, from 180-200 minutes to 30-45 minutes (83.3% time saving), and reduced noise in irrelevant user story datasets, so it is expected to increase the accuracy of the system, which will be tested in the evaluation section.

4.4. Finding Reusable User Stories

The fourth stage of this research is looking for reusable user stories. The search for user stories that can be reused is done by comparing each user story with the user story extracted from the application description. This stage is carried out in six steps: application description extraction, sentence tokenization, data cleaning, POS tagging, POS chunking, and calculating similarity.

4.4.1. Application Description Extraction

Application description will be extracted to obtain software features by tokenizing sentences, cleaning data, post tagging, heading chunking, case folding, word net lemmatization, and deletion of stopwords. Tokenization separates the application description, which is still one text, into one sentence for one need. Tokenize on the app description using the sent tokenize () function. Data cleaning is done because the application description contains many disturbing characters for the next stage. These characters include punctuation at the end of words and other irrelevant characters. Data cleaning is done by calling the remove_punctuation () function. POS tagging is an NLP function that automatically labels words in sentences. POS tagging uses the POS-tag function in the NLTK library for each review sentence iteration. Chunking is done to group words or phrases in a sentence into a new group called ‘chunks.’ Chunking is done by combining parts of speech with regular expressions. The rules or rules used for chunking are adjusted to what phrase or group of words needs to be taken from a sentence. In this study, the rules used to retrieve Software Features (SF) are contained in Equation (6), where to obtain SF rules, Noun Phrase (NC) rules are needed in Equation (3), Verbs (VERB) in Equation (4), and Nouns (NP) in Equation (5).

$$NC = \{(< DT >*< JJ|JJS|JJR >*)?\,(< IN >*< NN|NNP|NNS|NNPS >)*\} \tag{3}$$

$$VERB = \{< VB|VBG|VBZ|VBD|VBN|VBP > +< IN >*\} \tag{4}$$

$$NP = \{((< NC > +<,>)*< NC >*< CC > +)*< NC|IN > +\} \tag{5}$$

$$SF\!:\{(< VERB|(JJ|JJS|JJR)|NP > +< DT >* < NP|(RB|RBR|RBS)|(JJ|JJS|JJR) > +)+\} \tag{6}$$

4.4.2. Calculating Similarity

Similarity calculations are performed to determine which user stories are related to feature extraction from the application description. The data extracted from the user story will be calculated for its proximity to the results of the feature extraction data from the application description. Proximity is assessed using the Cosine similarity and Jaccard similarity methods. Suppose a user story has a similarity value with a software feature that exceeds a predetermined threshold. In that case, the user story is considered to be related to the features in the application description. User stories that have a connection will be displayed in the results of user stories that can be reused. To get the best results, trials were carried out using three thresholds for each similarity calculation method. Calculations were carried out using the Cosine similarity method with 70%, 80%, and 90%, and Jaccard similarity with 5%, 10%, and 15% thresholds. The results of similarity calculations can be seen in Table 4. The most considerable value in each method is given a bold effect. The purpose of making this system is to recommend to users which user stories can be used. The expected recommendation is that the data generated is quite a lot and varied but still has minimal errors to produce information that is credible and usable so that this system prioritizes recall but does not ignore the value of precision so that in this study, the F1-score is better used as a reference for finding the best method.

Based on the results from Table 4, the best method will refer to the average of the F1-score, where avg means the average of data that is 0 (irrelevant) and 1 (relevant). The F1-score is the harmonic average of the precision and recall, in which the more significant the F1-score means, the better the precision and recall results; vice versa, the smaller the F1-score means, the worse the precision and recall results.

Table 4. User story results that can be reused.

Project	Method	0 (not relevant)			1 (relevant)			Average		
		Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Recycle	Cosine (0,7)	1.00	0.95	0.98	0.24	1.00	0.38	0.62	0.98	0.68
	Cosine (0,8)	1.00	0.98	0.99	0.42	1.00	0.59	0.71	0.99	0.79
	Cosine (0,9)	0.99	1.00	0.99	1.00	0.00	0.00	0.99	0.50	0.50
	Jaccard (0,05)	1.00	0.01	0.02	0.01	1.00	0.03	0.51	0.50	0.02
	Jaccard (0,1)	0.87	0.04	0.08	0.01	0.58	0.02	0.44	0.31	0.05
	Jaccard (0,15)	0.83	0.05	0.10	0.00	0.21	0.01	0.41	0.13	0.05
Archive	Cosine (0,7)	0.95	0.87	0.91	0.45	0.67	0.54	0.70	0.77	0.72
	Cosine (0,8)	0.95	0.99	0.97	0.94	0.67	0.79	0.95	0.83	0.88
	Cosine (0,9)	0.87	1.00	0.93	1.00	0.01	0.02	0.93	0.50	0.47
	Jaccard (0,05)	0.75	0.05	0.09	0.13	0.90	0.22	0.44	0.47	0.16
	Jaccard (0,1)	0.62	0.13	0.22	0.08	0.47	0.13	0.35	0.30	0.18
	Jaccard (0,15)	0.63	0.17	0.26	0.06	0.35	0.10	0.34	0.26	0.18
Alfred	Cosine (0,7)	1.00	0.91	0.95	0.15	1.00	0.27	0.58	0.96	0.61
	Cosine (0,8)	1.00	1.00	1.00	0.76	0.96	0.85	0.88	0.98	0.92
	Cosine (0,9)	0.98	1.00	0.99	1.00	0.00	0.00	0.99	0.50	0.50
	Jaccard (0,05)	0.95	0.05	0.10	0.01	0.81	0.03	0.48	0.43	0.06
	Jaccard (0,1)	0.89	0.10	0.17	0.00	0.27	0.01	0.45	0.18	0.09
	Jaccard (0,15)	0.89	0.12	0.22	0.00	0.00	0.00	0.44	0.06	0.11

In the project recycle using the Cosine similarity method, it has the best value at the threshold of 0.8 with an average precision of 0.71, recall of 0.99, and F1-score of 0.79. The formula is used in the equations below to get this value. The first thing to do is to get each precision-recall and F1-score from irrelevant (0) and relevant (1) user stories. For irrelevant user stories with 1620 True Positives (TP), 0 False Positives (FP), and 33 False Negatives. For relevant user stories with 24 True Positives (TP), 32 False Positives (FP) and 0 False Negatives. After obtaining each precision, recall, and F1-score, a macro average search is performed, namely taking the average of each value. In the Cosine similarity method, with a threshold of 0.8, the best value is in the project archive, while at a threshold of 0.8, the best value is in the Alfred project; for a threshold of 0.9, the best value is in the recycle and Alfred projects. The Jaccard similarity method produces the project archive's best value with a 0.05, 0.1, 0.15 threshold.

4.5. Discussion

This research begins with data collection, determining the topic from the public user story dataset [9], which is carried out for the search process for application descriptions. The application descriptions are collected by searching on the Google search engine with predetermined topics. After obtaining the application description, an online meeting process was carried out using Google Meet with the evaluators. The meeting began with the researchers' explanation of the background and purpose of this study, then continued by dividing sheets using Google Docs containing user story datasets and application description source links. The evaluator's annotation process takes seven days. After seven days, the sheets were collected and then averaged. If more than half (50%) of the evaluators chose the user story, then the user story was considered relevant. At the same time, during the annotation period from the evaluator, the user story dataset and application description are pre-processed.

This study has a limitation: when searching for user stories related to software feature extraction, the system runs for over three hours and has a reasonably low evaluation result. This happens because the 22 user story datasets contain 1677 user stories compared to extraction software features of each application description. So, an initial filtration process is needed, which aims to find links between the application description and the user story dataset. The experiment was carried out using the three methods discussed in sub-chapter 3.2 with the result that the user story dataset is smaller in comparison, thereby increasing the evaluation value and shortening the processing time. The performance increase occurred because previously, the user story dataset, which was compared with the application description of 22, decreased to only two datasets on the recycle and Alfred topics and five datasets on the archive topic according to the results of Table 3 so that the data compared was less and reduced user stories irrelevant in the dataset. Feature extraction is performed on application descriptions to identify relevant software functionalities. However, application descriptions vary significantly in structure and detail, leading to potential inconsistencies in feature extraction. While our approach provides a structured method for extracting features, handling such variability remain challenging. Future work could explore fine-tuning transformer-based models or incorporating metadata-based heuristics to improve the consistency and accuracy of feature extraction across diverse application descriptions.

After the filtration stage is executed, the next step is to search for user stories that can be reused. This stage begins by extracting the software features contained in the application description. The application description will be pre-processed, and the tagging and chunking process will follow the specified rules. The results of this extraction will be searched for closeness to each user story that has gone through the filtration process. Experiments were carried out using two similarity calculation methods, namely Cosine and Jaccard similarity. The results of the closeness calculation will be evaluated using a precision and recall test with the annotation results from the evaluator. The method with the most significant precision and recall test values will be used at this stage. The evaluation results in Table 4 show that the Cosine similarity method with a threshold of 0.8 is the most appropriate method for this study, with F1-scores for each topic being 0.79, 0.88, and 0.92.

The results of this study indicate that the word extraction method uses noun phrase extraction, as done by Bakar *et al*. [2], combined with TF-IDF, can be used as a filtration step compared to other methods tested in this study, which were calculated using the precision and recall test. Cosine and Jaccard similarity are methods used to find closeness between two sentences [12]. In this research, the methods used for filtration are TF-IDF aimed at extracting frequently occurring words, combining TF-IDF with the extraction of nouns found in the user story dataset, and utilizing YAKE tools for automated word extraction. While our approach effectively identifies relevant user stories using Cosine similarity and keyword extraction, we recognize that more advanced NLP techniques, such as BERT-based embedding, can improve semantic matching [8, 19]. These models take advantage of deep contextual embedding, which can provide a deeper understanding of user stories. This study prioritizes TF-IDF and Cosine similarity due to their simplicity, interpretability, and lower computational cost. Future work may explore integrating transformer-based approaches to improve accuracy and assess their feasibility in practical software engineering applications.

This shows that this approach is practical for improving the reuse task of user stories based on feature extraction from application descriptions. One of the limitations of this study is that the number of different projects is relatively small (22). However, the dataset contains a large number of user stories (1,677), which provides a solid basis for pattern analysis. It is important to be aware of the limitations of our datasets. Future research should validate approaches on more diverse and broad data sets to ensure their generalization across multiple contexts. Therefore, the results presented in this study should be interpreted as indicative, not conclusive. The performance of the proposed method may differ when applied to projects from different domains or with different characteristics.

Given the diversity of topics in the user story dataset used in this research, it is further suggested to seek descriptions of other applications with different relevance from the designated topic in this study. This would lead to testing a broader range of user story datasets.

Reusing requirements has been shown to improve efficiency in software development [15, 22]. However, adapting reused user stories across multiple domains presents challenges due to terminology, context, and variations in functional requirements. For example, user stories from e-commerce may require significant modifications when applied to healthcare. The ease of adaptation depends on the similarities between the original and target app descriptions-a closer match simplifies the process, while significant differences require more effort. Common adaptations include terminology changes (e.g., “customer” in e-commerce vs. “patient” in healthcare), feature modifications (e.g., a “checkout” process in retail that focuses on payments vs. procurement that requires approval steps), and structural refinements (e.g., adding restrictions or details to fit the new domain). While these adjustments help maintain relevance, the system cannot function independently. The intervention of a system analyst is essential, as the system provides recommendations rather than fully automated solutions.

The study proposes an automated approach to filter and extract reusable user stories to improve efficiency in agile development. While our method improves automation, we realize that direct comparisons with traditional manual requirements collection or existing tools can further validate its effectiveness. However, we evaluated by involving experts and comparing the recommended reusable user stories with expert opinions. Future studies will explore such comparative analyses to measure time efficiency and accuracy improvements.

Several studies have investigated approaches to enhance user story processing and reuse. A taxonomy-based method has been introduced to support user story reuse, offering a different strategy from NLP-based techniques while aiming for similar improvements in reusability [7]. Additionally, prior research has emphasized the significance of NLP in automating user story processing, mainly through models utilizing pre-trained embeddings [23, 28]. This study aligns with these advancements by employing feature extraction and similarity measurements to identify reusable user stories, contributing to the ongoing development of NLP-driven requirements engineering. The proposed method improves user story reuse and can be integrated into agile workflows through backlog refinement, where similar past user stories are identified and suggested during requirement discussions [32]. However, adopting this approach in industry presents challenges, such as aligning with existing tools (e.g., Jira, Azure DevOps) and addressing variations in user story structuring across teams. Additionally, teams may require training to effectively utilize automated recommendations, and some stakeholders may be hesitant to rely on automation in requirement elicitation. Addressing these challenges will be crucial for the broader adoption of this approach.

5. Conclusions

The main contribution of this study is the development and validation of an NLP-based approach for reusing user stories from application descriptions in software requirement elicitation to enhance cost and time efficiency in software development. The findings of this study confirm that reusing user stories based on feature extraction from application descriptions using Cosine similarity with an 80% threshold is an effective method for improving time and cost efficiency in software development. The proposed NLP-based approach, which employs POS tagging and similarity measurement, demonstrated its feasibility by achieving high precision (84%), recall (93%), and F1-score (86%) across three test projects. Specifically, the method achieved precision, recall, and F1-scores of 0.71, 0.99, and 0.79 for Recycle; 0.95, 0.83, and 0.88 for Archive; and 0.88, 0.98, and 0.92 for Alfred, confirming its alignment with system design requirements. These results validate the effectiveness of the Cosine similarity method in identifying and reusing user stories, making it a viable solution for improving efficiency in the software requirement elicitation process.

Acknowledgment

This work was supported by Universitas Airlangga under Airlangga Research Fund (ARF) Batch 2 2025, Grand number 3005/B/UN3.LPPM/PT.01.03/2025.

References

[1] Ali F., Usman M., Abrar M., Rahman S., and et al., "Practices of Motivators in Adopting Agile Software Development at Large Scale Development Team from Management Perspective," *Electronics*, vol. 10, no. 19. 2021. DOI: 10.3390/electronics10192341

[2] Bakar N., Kasirun Z., Salleh N., and Jalab H., "Extracting Features from Online Software Reviews to Aid Requirements Reuse," *Applied Soft Computing*, vol. 49, pp. 1297-1315, 2016. DOI: 10.1016/j.asoc.2016.07.048

[3] Buglione L. and Abran A., "Improving the User Story Agile Technique Using the INVEST Criteria," *in Proceedings of the 23rd Joint Conference of International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, Ankara, pp. 49-53, 2013. DOI: 10.1109/IWSM-Mensura.2013.18

[4] Cico O., Jaccheri L., Nguyen-Duc A., and Zhang H., "Exploring the Intersection between Software Industry and Software Engineering Education-A Systematic Mapping of Software Engineering Trends," *Journal of Systems and Software*, vol. 172, pp. 110736, 2021. DOI: https://doi.org/10.1016/j.jss.2020.110736

[5] Dalpiaz F., Requirements Data Sets (User Stories), Mendeley Data, http://doi.org/10.17632/7zbk8zsd8y.1, Last Visited, 2024.

[6] Dalpiaz F. and Brinkkemper S., "Agile Requirements Engineering with User Stories," *in Proceedings of the IEEE 26th International Requirements Engineering Conference*, Banff, pp. 506-507, 2018. DOI: 10.1109/RE.2018.00075

[7] Dilorenzo E., Dantas E., Perkusich M., Ramos F., and et al., "Enabling the Reuse of Software Development Assets Through a Taxonomy for User Stories," *IEEE Access*, vol. 8, pp. 107285-107300, 2020. DOI: 10.1109/ACCESS.2020.2996951

[8] Fahmi F. and Pratiwi A.,, "Identifying Sentiment in User Reviews of Get Contact Application Using Natural Language Processing," *in Proceedings of the International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarang, pp. 428-433. DOI: 10.1109/iSemantic63362.2024.10762453

[9] Gunes T. and Aydemir F., "Automated Goal Model Extraction from User Stories Using NLP," *in Proceedings of the IEEE 28th International Requirements Engineering Conference*, Zurich, pp. 382-387, 2020. DOI: 10.1109/RE48521.2020.00052

[10] Halme E., Jantunen M., Vakkuri V., Kemell K., and Abrahamsson P., "Making Ethics Practical: User Stories as a Way of Implementing Ethical Consideration in Software Engineering," *Information and Software Technology*, vol. 167, pp. 1-37, 2024. DOI: 10.1016/j.infsof.2023.107379

[11] IEEE Standards Association, IEEE Standard for Information Technology-System and Software Life Cycle Processes-Reuse Processes, IEEE Stand, https://standards.ieee.org/ieee/1517/4603/, Last Visited, 2024.

[12] Jain A., Jain A., Chauhan N., Singh V., and Thakur N., "Information Retrieval Using Cosine and Jaccard Similarity Measures in Vector Space Model," *International Journal of Computer Applications*, vol. 164, no. 6, pp. 28-30, 2017. DOI: 10.5120/ijca2017913699

[13] Jiang H., Ma H., Ren Z., Zhang J., and Li X., "What Makes a Good App Description?," *in Proceedings of the 6th Asia-Pacific Symposium on Internetware on Internetware*, Hong Kong, pp. 45-53, 2014. DOI: 10.1145/2677832.2677842

[14] Johann T., Stanik C., Alizadeh A., and Maalej W., "SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews," *in Proceedings of the IEEE 25th International Requirements Engineering Conference*, Lisbon, pp. 21-30, 2017. DOI: 10.1109/RE.2017.71

[15] Khan M., Saadatmand M., Enoiu E., Sundamark D., and Lindskog C., *Automated Reuse Recommendation of Product Line Assets Based on Natural Language Requirements*, Reuse in Emerging Software Engineering Practices, 2020. https://doi.org/10.1007/978-3-030-64694-3_11

[16] Kuhrmann M., Tell P., Hebig R., Klunder J., and et al., "What Makes Agile Software Development Agile?," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3523-3539, 2022. DOI: 10.1109/TSE.2021.3099532

[17] Lee W. and Chen C., "Agile Software Development and Reuse Approach with Scrum and Software Product Line Engineering," *Electronics*, vol. 12, no. 15, pp. 3291, 2023. DOI: 10.3390/electronics12153291

[18] Lucassen G., Dalpiaz F., Werf J., and Brinkkemper S., "Improving Agile Requirements: The Quality User Story Framework and Tool," *Requirements Engineering*, vol. 21, no. 3, pp. 383-403, 2016. DOI: 10.1007/s00766-016-0250-x

[19] Molla Y., Alemneh E., and Yimer S., "COSMIC-Based Early Software Size Estimation Using Deep Learning and Domain-Specific BERT," *IEEE Access*, vol. 13, pp. 28463-28475, 2025. DOI: 10.1109/ACCESS.2025.3540548

[20] Muhamad F., Hamid S., Subramaniam H., Abdul Rashid R., and Fahmi F., "Fault-Prone Software Requirements Specification Detection Using Ensemble Learning for Edge/Cloud Applications," *Applied Sciences*, vol. 13, no. 14, pp. 8368, 2023. DOI: 10.3390/app13148368

[21] O'hEocha C. and Conboy K., "The Role of the User Story Agile Practice in Innovation," *Lecture Notes in Business Information Processing*, vol. 65, pp. 20-30, 2010. DOI: 10.1007/978-3-642-16416-3_3

[22] Pirzadeh H., Oliveira A., and Shanian S., "ReUse: A Recommendation System for Implementing User Stories," *in Proceedings of the 11th International Conference on Software Engineering Advances*, Montreal, pp. 149-153, 2016. file:///C:/Users/acit2k/Downloads/icsea_2016_6_20_10100%20(1).pdf

[23] Raharjana I., Siahaan D., and Fatichah C., "User Stories and Natural Language Processing: A Systematic Literature Review," *IEEE Access*, vol. 9, pp. 53811-53826, 2021. DOI: 10.1109/ACCESS.2021.3070606

[24] Resketi M., Motameni H., Nematzadeh H., and Akbari E., "Automatic Summarising of User Stories in Order to be Reused in Future Similar Projects," *IET Software*, vol. 14, no. 6, pp. 711-723, 2020. DOI: 10.1049/iet-sen.2019.0182

[25] Sabahat N., Iqbal F., Azam F., and Javed M., "An Iterative Approach for Global Requirements Elicitation: A Case Study Analysis," *in Proceedings of the International Conference on Electronics and Information Engineering*, Kyoto, pp. 361-366, 2010. DOI: 10.1109/ICEIE.2010.5559859

[26] Salazar G., Mora M., Limon H., Rodriguez F., and Zavala A., "Review of Agile SDLC for Big Data Analytics Systems in the Context of Small Organizations Using Scrum-XP," *The International Arab Journal of Information Technology*, vol. 21, no. 6, pp. 1089-1110, 2024. DOI: 10.34028/iajit/21/6/12

[27] Schots M., "On the Use of Visualization for Supporting Software Reuse," *in Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, pp. 694-697, 2014. DOI: 10.1145/2591062.2591095

[28] Scoggin S. and Neto H., "Identifying Valid User Stories Using BERT Pre-Trained Natural Language Models," *Information Systems and Technologies*, pp. 167-177, 2024. https://doi.org/10.1007/978-3-031-45648-0_17

[29] Sharma S. and Pandey S., "Requirements Elicitation: Issues and Challenges," *in Proceedings of the International Conference on Computing for Sustainable Global Development*, New Delhi, pp. 151-155, 2014. DOI: 10.1109/IndiaCom.2014.6828119

[30] Siahaan D., Raharjana I., and Fatichah C., "User Story Extraction from Natural Language for Requirements Elicitation: Identify Software-Related Information from Online News," *Information and Software Technology*, vol. 158, pp. 107195, 2023. DOI: 10.1016/j.infsof.2023.107195

[31] Suali A., Fauzi S., Nasir M., Sobri W., and Raharjana I., "Software Quality Measurement in Software Engineering Project: A Systematic Literature Review," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 3, pp. 918-929, 2019. https://www.jatit.org/volumes/Vol97No3/18Vol97No3.pdf

[32] Suganya R., Banerjee D., Mishra A., Subbulakshmi T., and Subramanian G., "Enhancing Agile Development in Tech Companies: Backlog Management, Tool Integration, and Stakeholder Collaboration," *in Proceedings of the 6th International Conference on Recent Trends in Advance Computing*, Chennai, pp. 718-724, 2023. DOI: 10.1109/ICRTAC59277.2023.10480864

[33] Tam C., Moura E., Oliveira T., and Varajão J., "The Factors Influencing the Success of on-Going Agile Software Development Projects," *International Journal of Project Management*, vol. 38, no. 3, pp. 165-176, 2020. DOI: 10.1016/j.ijproman.2020.02.001

[34] Tariq S., Ibrahim A., Usama A., and Shahbaz M.,

"An Overview of Requirements Elicitation Techniques in Software Engineering with a Focus on Agile Development," *in Proceedings of the 4th International Conference on Computing and Information Sciences*, Karachi, pp. 1-6, 2021. DOI: 10.1109/ICCIS54243.2021.9676192

[35] Thamrongchote C. and Vatanawood W., "Business Process Ontology for Defining User Story," *in Proceedings of the IEEE/ACIS 15th International Conference on Computer and Information Science*, Okayama, pp. 3-6, 2016. DOI: 10.1109/ICIS.2016.7550829

[36] Trisnawati E., Raharjana I., Taufik T., Basori A., and et al., "Analyzing Variances in User Story Characteristics : A Comparative Study of Stakeholders with Diverse Domain and Technical Knowledge in Software Requirements Elicitation," *Information Systems Engineering and Business Intelligence*, vol. 10, no. 1, pp. 110-125, 2024. DOI: 10.20473/jisebi.10.1.110-125

[37] Wang Y., Wang J., Zhang H., Ming X., and et al., "Where is your App Frustrating Users?," *in Proceedings of the 44th International Conference on Software Engineering*, New York, pp. 2427-2439, 2022. DOI: 10.1145/3510003.3510189

[38] Wu H., Deng W., Niu X., and Nie C., "Identifying Key Features from App User Reviews," *in Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*, Madrid, pp. 922-932, 2021. DOI: 10.1109/ICSE43902.2021.00088.

[39] Zainal D., Razali R., and Mansor Z., "Team Formation for Agile Software Development-Crowdsourcing-based Empirical Study," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 34, no. 2, pp. 133-143, 2024. DOI: 10.37934/araset.34.2.133143

[40] Zowghi D. and Coulin C., *Engineering and Managing Software Requirements*, Springer Nature, 2005. https://doi.org/10.1007/3-540-28244-0_2

Indra Kharisma Raharjana is an Associate Professor at Universitas Airlangga. He holds degrees from Institut Teknologi Sepuluh Nopember and Institut Teknologi Bandung. His research focuses on software engineering and natural language processing. He is Editor-in-Chief of the Journal of Information Systems Engineering and Business Intelligence and an active member of CASE, IEEE, ACM, INACL, and AISINDO.

Avril Hermawan received a bachelor's degree in information systems from the Universitas Airlangga, Indonesia, in 2023. He currently works as a digital product campaign specialist at PT. Bank Tabungan Negara (BTN). His professional interests include Digital Marketing, Digital Product Management, and User Experience.

Badrus Zaman is a lecturer in the information systems study program at Universitas Airlangga. He received a bachelor's degree in informatics engineering from the Institut Teknologi Sepuluh Nopember, Indonesia, in 2005, and a master's degree in computer science from the Universitas Gadjah Mada, Indonesia, in 2011. His research interests Cover Information Systems and Natural Language Processing.

Shukor Sanim Mohd Fauzi is a Senior Lecturer at the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA. He holds degrees from UiTM, Universiti Teknologi Malaysia, and a Ph.D. in Software Engineering from UNSW. His research interests include Software Engineering, Empirical Studies, Software Repository Mining, Social Network Analysis, Socio-Technical Congruence, and Collaborative Software Processes.