



LEVERAGING TEXT GENERATION FOR ENHANCED USER STORY QUALITY

PAR CARLOS ALBERTO DOS SANTOS

**THESIS PRESENTED TO L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI IN
PARTIAL FULFILLMENT OF THE REQUIERMENTS FOR THE DEGREE OF
PHILOSOPHIÆ DOCTOR (PH.D.) IN THE SUBJECT OF INFORMATIQUE**

QUÉBEC, CANADA

© CARLOS ALBERTO DOS SANTOS, 2025

ABSTRACT

Context: User stories play a crucial role in agile software development because of their structured format and ease of implementation. However, development teams face the challenging task of managing the variety of information required from multiple sources to craft user stories manually. Furthermore, poor-quality user stories can hinder communication among team members, potentially causing delays or leading to errors in the development process. **Goal:** This thesis investigates the state-of-the-art in the automatic generation of user stories and proposes various text generation models to assist in writing user stories within Agile Software Development (ASD) projects. We hypothesize that employing these models can help software practitioners write user stories more efficiently and with improved quality. **Method:** A range of research methods were used to construct and evaluate this thesis. Firstly, we conducted a Systematic Literature Review (SLR) to summarize the evidence on the topic. Based on the findings of our SLR, we introduced our two first text generation models for user stories (N-gram and GPT) and used a quantitative framework of metrics to compare them. Subsequently, we improved the N-gram model created and performed a controlled experiment followed by a survey designed to evaluate the use of text generation models for user stories. **Results:** The SLR found that there is a shortage of user stories corpora to support the implementation of text generation models for user stories as well as a wide variety of different Natural Language Processing (NLP) techniques and Machine Learning (ML) algorithms to specify user stories automatically. Only a few studies are concerned about the quality of the user stories generated by the approaches presented. Quantitative evaluation of the initial N-gram model using BLEU, ROUGE, and BERTScore metrics showed that while GPT models excelled in developing more comprehensive user stories, N-gram models demonstrated a higher degree of semantic sensitivity. Finally, our controlled experiment revealed that the upgraded version of the N-gram model enhanced the *consistency* and *uniformity* of the user stories compared to the manual writing method and brought important insights about user stories employment. **Conclusion:** The use of text generation models for supporting the writing of user stories is promising. These models accelerated the composition process and improved the quality of the resulting user stories. We encourage further investigations in the direction of refining the N-gram models technique or training other Large Language Models (LLMs) to support the writing of user stories in different contexts with varied templates.

RÉSUMÉ

Contexte: Les récits d'utilisateurs jouent un rôle crucial dans le développement de logiciels agiles en raison de leur format structuré et de leur facilité de mise en œuvre. Cependant, les équipes de développement sont confrontées à la tâche difficile de gérer la variété des informations requises à partir de sources multiples pour rédiger manuellement les récits d'utilisateurs. En outre, des récits d'utilisateur de mauvaise qualité peuvent entraver la communication entre les membres de l'équipe, ce qui peut entraîner des retards ou des erreurs dans le processus de développement. **Objectif:** Cette thèse étudie l'état de l'art dans la génération automatique d'histoires d'utilisateurs et propose divers modèles de génération de texte pour aider à la rédaction d'histoires d'utilisateurs dans le cadre de projets de développement de logiciels agiles. Nous émettons l'hypothèse que l'utilisation de ces modèles peut aider les praticiens du logiciel à écrire des histoires d'utilisateurs plus efficacement et avec une meilleure qualité. **Méthode:** Plusieurs méthodes de recherche ont été utilisées pour élaborer et évaluer cette thèse. Tout d'abord, nous avons effectué une revue systématique de la littérature pour résumer les preuves sur le sujet. Sur la base des résultats de cette analyse, nous avons présenté nos deux premiers modèles de génération de texte pour les récits d'utilisateurs (N-gram et GPT) et nous avons utilisé un cadre quantitatif de mesures pour les comparer. Par la suite, nous avons amélioré le modèle N-gram créé et réalisé une expérience contrôlée suivie d'une enquête destinée à évaluer l'utilisation des modèles de génération de texte pour les récits d'utilisateurs. **Résultats:** La revue a constaté qu'il y a une pénurie de corpus de récits d'utilisateurs pour soutenir la mise en œuvre de modèles de génération de texte pour les récits d'utilisateurs, ainsi qu'une grande variété de techniques de traitement du langage naturel et d'algorithmes d'apprentissage automatique pour spécifier automatiquement les récits d'utilisateurs. Seules quelques études s'intéressent à la qualité des récits d'utilisateurs générés par les approches présentées. L'évaluation quantitative du modèle N-gram initial à l'aide des métriques BLEU, ROUGE et BERTScore a montré que si les modèles GPT ont excellé dans l'élaboration de récits d'utilisateurs plus complets, les modèles N-gram ont fait preuve d'un degré plus élevé de sensibilité sémantique. Enfin, notre expérience contrôlée a révélé que la version améliorée du modèle N-gram améliorerait la *consistance* et la *uniformité* des récits d'utilisateurs par rapport à la méthode de rédaction manuelle et apportait des informations importantes sur l'emploi des récits d'utilisateurs. **Conclusion:** L'utilisation de modèles de génération de texte pour soutenir la rédaction d'histoires d'utilisateurs est prometteuse. Ces modèles ont accéléré le processus de composition et amélioré la qualité des récits d'utilisateurs qui en résultent. Nous encourageons la poursuite des recherches en vue d'affiner la technique des modèles N-grammes ou d'entraîner d'autres grands modèles de langage pour aider à la rédaction de récits d'utilisateurs dans différents contextes avec des modèles variés.

TABLE OF CONTENTS

ABSTRACT	ii
RÉSUMÉ	iii
LIST OF TABLES	vii
LIST OF FIGURES	ix
LIST OF ABBREVIATIONS	x
ACKNOWLEDGEMENTS	xi
INTRODUCTION	1
CHAPTER I – BACKGROUND	9
1.1 REQUIREMENTS ENGINEERING	10
1.1.1 REQUIREMENTS IN AGILE SOFTWARE DEVELOPMENT	13
1.1.2 REQUIREMENTS SPECIFICATION IN PRACTICE	20
1.2 REQUIREMENTS QUALITY	23
1.2.1 USER STORIES QUALITY	26
1.3 TEXT PREDICTION	32
1.3.1 PROBABILISTIC METHODS	33
1.3.2 DEEP LEARNING	36
1.3.3 QUANTITATIVE METRICS	40
1.4 CONCLUSION	42
CHAPTER II – A COMPREHENSIVE SYSTEMATIC LITERATURE RE- VIEW ABOUT AUTOMATIC USER STORY GENERATION	44
2.1 INTRODUCTION	44
2.1.1 SIMILAR REVIEWS	44
2.2 RESEARCH DESIGN	48
2.2.1 SEARCH STRATEGY AND STUDY SELECTION	49
2.2.2 QUALITY ASSESSMENT	52

2.2.3	DATA EXTRACTION AND ANALYSIS	54
2.3	RESULTS	56
2.3.1	SUMMARY OF STUDIES	56
2.3.2	(RQ1) WHAT DATASETS ARE USED BY RESEARCHERS, AND IN WHAT WAY, TO GENERATE USER STORIES AUTOMATICALLY? .	57
2.3.3	(RQ2) WHICH TECHNIQUES ARE USED TO SPECIFY USER STO- RIES AUTOMATICALLY?	61
2.3.4	(RQ3) HOW USER STORY QUALITY IS EVALUATED?	68
2.4	DISCUSSION	71
2.4.1	RQ1 - DISCUSSING THE AVAILABILITY OF USER STORIES DATASETS	72
2.4.2	RQ2 - DISCUSSING THE APPROACHES EXPLORED BY THE PRI- MARY STUDIES	74
2.4.3	RQ3 - DISCUSSING THE EMPLOYMENT OF USER STORY QUAL- ITY GUIDELINES	82
2.5	FUTURE WORK	85
2.6	THREATS TO VALIDITY	87
2.7	CONCLUSION	89
CHAPTER III – AI-DRIVEN USER STORY GENERATION		91
3.1	INTRODUCTION	91
3.2	USER STORY GENERATION	92
3.2.1	N-GRAM MODEL	92
3.2.2	GPT MODEL	94
3.3	EVALUATION	95
3.3.1	N-GRAM MODEL	95
3.3.2	GPT MODEL	98
3.4	DISCUSSION	100
3.5	CONCLUSION	102
CHAPTER IV – LEVERAGING TEXT GENERATION FOR ENHANCED USER STORY QUALITY: A CONTROLLED EXPERIMENT		103

4.1	INTRODUCTION	103
4.2	ADVANCING USER STORY TEXT GENERATION WITH N-GRAM MODELS	103
4.2.1	PROTOTYPE TOOL	109
4.3	METHODS	110
4.3.1	INCLUSION CRITERIONS	111
4.3.2	EXPERIMENT DESIGN	112
4.3.3	VALIDATION	114
4.3.4	SURVEY	115
4.4	RESULTS	116
4.4.1	CONTROLLED EXPERIMENT	117
4.4.2	SURVEY - CONTROL GROUP	120
4.4.3	SURVEY - EXPERIMENTAL GROUP	123
4.5	DISCUSSION	127
4.5.1	SURVEY	129
4.5.2	LIMITATIONS	130
4.6	THREATS TO VALIDITY	131
4.7	CONCLUSION	133
	CONCLUSION	135
	REFERENCES	142
	APPENDIX A – ETHICS CERTIFICATION	158

LIST OF TABLES

TABLE 1.1 : USE CASE SCENARIO EXAMPLE EXTRACTED FROM A PROJECT OF SYSTEM INTEGRATION OF A TRANSPORTATION COMPANY (AUTHOR’S PERSONAL ARCHIVE).	16
TABLE 1.2 : 13 LANGUAGE CRITERIA OF THE QUALITY USER STORY (QUS) FRAMEWORK (ADAPTED FROM Lucassen et al. (2015))..	29
TABLE 1.3 : BIGRAM PROBABILITIES FOR EIGHT WORDS EXTRACT FROM A CORPUS. Jurafsky & Martin (2009)	35
TABLE 1.4 : SUMMARY EXPLANATION OF THE TEXT PREDICTION METRICS USED IN THIS RESEARCH.	41
TABLE 2.1 : SUMMARY OF RELATED STUDIES.. . . .	47
TABLE 2.2 : Research questions and their rationales.. . . .	48
TABLE 2.3 : Quality assessment questions Wieringa et al. (2006)	53
TABLE 2.4 : Quality assessment scores.	54
TABLE 2.5 : Data extraction form.. . . .	55
TABLE 2.6 : Primary studies and the datasets used to automatically identify and/or generate user stories.. . . .	58
TABLE 2.7 : Main techniques used for each primary study to identify and generate user stories automatically.	61
TABLE 2.8 : NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING TECHNIQUES USED IN PRIMARY STUDIES TO IDENTIFY AND GENERATE USER STORIES AUTOMATICALLY (NLP = NATURAL LANGUAGE PROCESSING, SL = SUPERVISED LEARNING, UL = UNSUPERVISED LEARNING).	67
TABLE 2.9 : Summary on how the quality of user story generated is evaluated by the primary studies. The symbol “-” means <i>none</i> or <i>unspecified</i>	68
TABLE 3.1 : QUANTITATIVE METRICS FOR N-GRAM MODEL.	98
TABLE 3.2 : QUANTITATIVE METRICS FOR GPT MODEL.	100

TABLE 4.1 :	N-GRAMS CREATED BASED ON LANGUAGE HEURISTICS TO MIMIC THE USER STORY TEMPLATE..	106
TABLE 4.2 :	N-GRAMS FOUND FOR INPUT SEARCH "AS A".	109
TABLE 4.3 :	LIST OF QUESTIONS AND THEIR RATIONALE PREPARED FOR THE CONTROL GROUP..	116
TABLE 4.4 :	LIST OF QUESTIONS AND THEIR RATIONALE PREPARED FOR THE EXPERIMENTAL GROUP.	117
TABLE 4.5 :	RESULTS AFTER RUNNING THE AQUASA TOOL IN THE USER STORIES PRODUCED BY BOTH STUDY GROUPS: EXPERIMENTAL AND CONTROL. THE TOOL EVALUATED ALL SYNTACTIC CRITERIA LISTED IN TABLE 1.2 AND PART OF THE PRAGMATIC ATTRIBUTES (<i>EXPLICIT DEPENDENCIES, UNIFORM AND UNIQUE</i>). 118	
TABLE 4.6 :	SUMMARY OF THE CONTROL GROUP SURVEY RESULTS OF 8 PARTICIPANTS. Q3 IS AN OPEN QUESTION ASKING PARTICIPANTS TO SUMMARIZE THE MAIN CHALLENGES WHEN WRITING USER STORIES..	122
TABLE 4.7 :	ISSUES REPORTED IN Q3 BY THE CONTROL GROUP.	122
TABLE 4.8 :	SUMMARY OF SURVEY RESULTS OF THE EXPERIMENTAL GROUP OF 8 PARTICIPANTS, CAPTURING RESPONSES TO 4 QUESTIONS AS PART OF THE STUDY'S EVALUATION PROCESS. Q4 IS A OPEN QUESTION ASKING PARTICIPANTS TO SUMMARIZE THE MAIN CHALLENGES WHEN WRITING USER STORIES. SOME PARTICIPANTS REPLIED TO THE Q4 MORE BROADLY, AND OTHERS RESTRICTED TO THE EXPERIMENT..	124
TABLE 4.9 :	ISSUES REPORTED BY THE EXPERIMENTAL GROUP RELATED TO Q4. SOME PARTICIPANTS REPLIED TO THE QUESTION MORE BROADLY, WHILE OTHERS SHARED THEIR EXPERIENCE USING OUR PROTOTYPE. *THE LAST TWO ISSUES ARE RELATED TO OUR PROTOTYPE..	126

LIST OF FIGURES

FIGURE 1.1 – WATERFALL MODEL FOR SOFTWARE DEVELOPMENT. THE WATERFALL MODEL IS NOT IDEAL WHEN TEAMS CAN COMMUNICATE INFORMALLY AND SOFTWARE REQUIREMENTS CHANGE FREQUENTLY. IN SUCH CASES, ITERATIVE DEVELOPMENT AND AGILE METHODS ARE MORE SUITABLE (ADAPTED FROM Sommerville (2016)).	11
FIGURE 1.2 – SUBDISCIPLINES OF SOFTWARE REQUIREMENTS ENGINEERING - ADAPTED FROM Wieggers & Beatty (2013) .	12
FIGURE 1.3 – AGILE DEVELOPMENT PROCESS ITERATION (ADAPTED FROM Pham (2018)).	14
FIGURE 1.4 – PHASES OF THE SPRING METHOD TO VALIDATE NEW SOFTWARE PRODUCTS. ON MONDAY, THE PARTICIPANTS MAP OUT THE PROBLEM AND CHOOSE A TARGET TO KEEP THE FOCUS. ON TUESDAY, THEY BRAINSTORM AND SKETCH NEW AND EXISTING IDEAS ON PAPER. ON WEDNESDAY, SOME HYPOTHESES ARE SET TO TEST THE IDEAS CREATED THE DAY BEFORE. ON THURSDAY, THE PARTICIPANTS CREATE REALISTIC PROTOTYPES, AND ON FRIDAY, THEY TEST THE PROTOTYPES WITH TARGET CUSTOMERS. Knapp et al. (2016) .	17
FIGURE 1.5 – TAXONOMY OF NEXT WORD PREDICTION SYSTEMS Hamarashid et al. (2022) .	32
FIGURE 2.1 – Search strategy and selection process.	51
FIGURE 2.2 – THE NUMBER OF COLLECTED CONFERENCE AND JOURNAL PAPERS UNTIL APRIL 2024.	57
FIGURE 3.1 – WORKFLOW DIAGRAM OF THE EXPERIMENTAL PROTOCOL.	93
FIGURE 4.1 – WORKFLOW DIAGRAM PRESENTING THE CREATION OF THE N-GRAM MODEL AND THE PROTOTYPE TOOL.	104
FIGURE 4.2 – PROTOTYPE PRESENTING THE ASSISTANCE TOOL POWERED BY OUR N-GRAM MODEL.	110
FIGURE 4.3 – WORKFLOW DIAGRAM PRESENTING THE STEPS OF OUR EXPERIMENT WITH THE EXPECTED OUTCOMES.	111

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AQUSA	Automatic Quality User Story Artisan
ASD	Agile Software Development
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
CPRE	Certified Professional for Requirements Engineering
GSD	Global Software Development
GPT	Generative Pre-trained Transformer
IREB	Requirements Engineering Board
LLM	Large Language Models
LSTM	Long Short Term Memory Networks
ML	Machine Learning
NLP	Natural Language Processing
OOSE	Object-Oriented Software Engineering
QUS	Quality User Story
RE	Requirements Engineering
RNN	Recurrent Neural Networks
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SE	Software Engineering
SLR	Systematic Literature Review
SRL	Semantic Role Labelling
TNN	Transformer Neural Networks

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for the health, intelligence and ability to complete this journey. My faith was an incredible reassurance during this learning process.

I would like to express my gratitude to my academic advisor, Kévin Bouchard for the outstanding support over this period. Thank you for trusting in my ability and providing me with the space I needed to produce this work. I would also like to thank my previous advisor Fábio Petrillo for the opportunities that were given.

A huge thank you to all my family in Brazil. For all the phone calls, prayers and visits that gave me the energy to keep going. Thank you for all the love you give me. Hug them all and be sure that this love is reciprocated. A special thank you to my parents João Santos and Ivanete Santos for giving me the gift of life and for always believing in my abilities.

I would like to extend my deepest gratitude to my wife, Débora Caetano De Bortoli, for all her support, unconditional love and loyalty during this time. Without her support this work would not have been possible. Also, to my loyal dog Zoe for being my emotional support many times. I Love you two!

To all my friends in Canada and Brazil. Thank you for listening to me, for encouraging me, for laughing with me in happy and sad times. You are my foundation and you can always count on me for the rest of your lives.

Lastly, I would like to thank everyone who, in some way, was part of my Ph.D. journey.

INTRODUCTION

Over the last decades, the growing complexity of software systems has pushed software developers to tackle not just technical challenges but also human, organizational, social, and political matters (Sommerville *et al.*, 2012). Agile Software Development (ASD) (Beck *et al.*, 2001) has emerged as a response to these challenges. At the heart of ASD lies iterative planning, enabling development teams to refine requirements incrementally and encouraging them to engage with stakeholders. Progress is managed among team members through daily stand-up meetings, and iterative client reviews help obtain feedback about the software product in development.

Adopting the agile process has encouraged development teams to engage with the business area. This also means that ASD embraces requirements engineering activities iteratively. Hence, elicitation, specification, documentation, and validation occur collaboratively and continuously throughout each development cycle (Ramesh *et al.*, 2010). While software requirements must be unambiguous, testable and agreed upon with the customer, ASD emphasizes a complete, consistent, and feasible collection of requirements (Bourque & Fairley, 2014).

In this context, ASD leverages user stories, which represent practical scenarios experienced by end users. They do not mean to be a requirement document but a reminder for future refinement (Shore & Warden, 2021). Cohn (2004) states that user stories are usually based on conversations between team members and stakeholders. They can also be written based on requirements artifacts, such as project documents, standards, business rules, or emails. Regardless of the source, user stories employ a semi-structured template to express a functionality valuable to the user. Using a template contributes to productivity and work deliverable quality (Dalpiaz *et al.*, 2019).

Schön *et al.* (2017) performed a systematic review in agile requirements engineering and found that user stories are the most frequently used artifact in ASD. Wagner *et al.* (2019) ran empirical research about the *status quo* in RE and concluded that free-form and constraint textual requirements (e.g. user stories) are sufficient for agile projects acting as reminders for further discussions. The textual free-form with no constraints means a text freely written without any template. In contrast, the constraint textual style involves relying on a specific and straightforward template to express the software requirements (Wagner *et al.*, 2019). User stories can have a defined template (e.g. “As a..., I want to..., so that...”) or even be written in a free-form way to speed up the writing process.

PROBLEM STATEMENT AND RESEARCH MOTIVATION

Agile processes can help technical teams keep the development focus on the business, deliver software that fits the purpose, and enable software evolution. To be closer to stakeholders, those teams take ownership of requirements refinement meetings, client communication and user story writing. However, bad requirements statements do not guide the development of good software products. In ASD, the client’s participation during the requirements review step is essential to ensure the quality of the requirements statements as it depends on the domain knowledge (Femmer *et al.*, 2017). This said, when the user’s needs are poorly written, they can communicate inaccurate messages to the development team, affecting further refinement meetings, coding and the delivery of the right software product.

Despite its empirical relevance (Lucassen *et al.*, 2016b; Kannan *et al.*, 2019), user stories are prone to ambiguity (Amna & Poels, 2022a). The variety of data sources and textual formats used as input to RE challenges software development teams (Aranda *et al.*, 2010). They need to read and comprehend the client’s needs from different sources and convert them into user stories or any other requirement specification format. This process is usually carried out

manually, demanding significant time and effort from the team in each software development iteration. [Fernández *et al.* \(2017\)](#) found that the most cited causes of failures in Requirements Elicitation (RE) are related to lack of experience and weak qualification of RE team members, lack of time, and missing completeness check of requirements.

These empirical studies collaborate to build the scenario of software requirements documentation nowadays:

- Software development projects based on agile methodology rarely follow formal standards of requirements documentation. Free-form writing or structured requirements lists with minimal constraints, such as user stories, are the most used formats for requirements documentation.
- User stories are lightweight documentation adopted in ASD environments. Even if it is not a formal way to document requirements, they can affect the subsequent steps of software development when they are ambiguous.
- Incomplete and/or hidden requirements is the most cited problem in RE, resulting in project delivery delays, rework and poor product quality; requirements inconsistency is cited as the cause of project failures.
- The most cited causes of incomplete and/or hidden requirements are the weak qualification/experience of the development team in RE, the lack of time and missing completeness check of requirements or procedures to deal with requirements documentation.
- Empirically, the usage of free-form writing and minimal constraints for requirements documentation, in addition to the lack of experience in Requirements Engineering (RE), can guide development teams to the risk of poor quality requirements documentation,

resulting in ambiguity, incompleteness and inconsistency, which are going to affect the project continuity.

RESEARCH GOALS

User stories are employed in the early stages of Agile Software Development (ASD) to gather the user's perspective about the system and guide conversations between stakeholders and the development team (Cohn, 2004). User stories can be written based on customer requirements but can also add or remove software requirements to the backlog after the proper refinement and alignment with stakeholders. They act as an interface between the development team and stakeholders, essential in requirements elicitation and specification in ASD.

Adopting automation in ASD is not a novelty and has been used in diverse subareas. There are approaches (Rodeghero *et al.*, 2017; Raharjana *et al.*, 2019; Peña Veitía *et al.*, 2020) investing in automation to support user story elicitation and specification in ASD environments. It was found that these studies do not use strict guidelines to assess the quality of the user stories generated by their approaches.

Therefore, in response to this scenario, this research proposes using automated text generation to assist software practitioners in writing user stories *faster* and *better*.

Consequently, these are the following goals of this research:

- **RG1 - Investigation of the state-of-the-art in user story generation**

This research aims to investigate which AI techniques have been used to generate user stories automatically, which datasets could be used to train AI models in this context, and how researchers have been evaluating the quality of the user stories generated by their models.

- **RG2 - Build a text prediction model able to assist the writing of user stories**

Based on the results gathered from the investigation of the state-of-the-art in user story generation, this research will prepare and evaluate its text prediction model using quantitative metrics.

- **RG3 - Perform a controlled experiment followed by a survey to assess the quality of the user stories written assisted by our text prediction model**

After preparing a text prediction model, the next step is to evaluate its performance qualitatively through a controlled experiment with software practitioners. This will be followed by a survey to test the concept's acceptability.

CONTRIBUTIONS

The concept of the user story generation supported by Artificial Intelligence (AI) models brings a new standard for ASD in Software Engineering (SE). This research found that automating user story writing streamlines the requirements specification process, boosting productivity and improving the quality and clarity of the stories, consequently leading to better software products.

In summary, this thesis contributes the following publications to the SE community:

- A Systematic Literature Review (SLR) about automatic user stories generation in the context of SE presenting the state-of-the-art in the topic. This study ([dos Santos et al., 2024](#)) was published in the *International Journal of Data Science and Analytics* in June 2024, entitled *Automatic user story generation: a comprehensive systematic literature review* ([DOI 10.1007/s41060-024-00567-0](#)).

- A conference paper presenting two approaches to automatically generate user stories and the method used for quantitatively evaluating their performance. This study ([Dos Santos et al., 2024](#)) was published in the *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)* in February 2024, entitled *AI-Driven User Story Generation* ([DOI 10.1109/ACDSA59508.2024.10467677](#)).
- A journal paper submitted to the *Requirements Engineering Journal* in January 2025 containing a refined approach to user story generation and the results obtained through the controlled experiment instrumented to assess the performance of the improved model.

In a most theoretic way, this thesis contributes with the following:

- Describing the foundation theory for user story generation, containing the topics ASD, RE and text prediction models (Chapter 1).
- Presenting different approaches employing NLP n-gram models to generate user stories automatically (Chapters 3 and 4).
- Demonstrating the method of fine-tuning a GPT model to generate text for user story statements (Chapter 3).
- Combining different NLP metrics to evaluate quantitatively the text generation models created (Chapter 3).
- Employing BERTScore as a first-hand qualitative measure before evaluating text prediction models with software practitioners (Chapter 3).

- Presenting a qualitative evaluation framework for assessing the refined n-gram text prediction model. The framework is based on a controlled experiment followed by a survey (Chapter 4).
- Showcasing the usage of the AQUASA tool to qualitatively evaluate the user stories written by the experiment participants supported by our text prediction model as part of the assessment process (Chapter 4).
- Collecting feedback from software practitioners about the text generation approach proposed and confirming their interest in the topic (Chapter 4).

THESIS OUTLINE

The remainder of this thesis is organized as follows:

- **Chapter 1** provides a comprehensive overview of the fundamental concepts and principles that form the basis of this research thesis, providing the necessary context to understand the study's objectives and methodology.
- **Chapter 2** introduces the SLR about automatic user stories generation. In this study, there were identified which AI techniques have been employed to support the generation of user stories in the context of SE, which datasets have been used for training AI models, how authors evaluated the quality of the user stories generated by their approaches; the SLR conducted served as a foundation for this thesis. This review provided a comprehensive understanding of the state-of-the-art methodologies, key findings, and prevailing gaps in the field. The insights gained from this review guided the design of the methodology and the development of the proposed solution. This chapter was mostly based on the SLR ([dos Santos et al., 2024](#)) published about user story generation.

- **Chapter 3** presents the two first approaches to generating user stories automatically: a pure NLP n-gram model powered by linguistic heuristics and a GPT fine-tuned model. Both models were evaluated quantitatively using NLP metrics for text prediction assessment. As a result, considering its simplicity and minimal processing requirements, the study recommended employing the n-gram technique for user story generation; this chapter was mainly based on the conference paper (Dos Santos *et al.*, 2024) published.
- **Chapter 4** exposes an improved text generation approach based on the n-gram model outlined in the previous study. A controlled experiment followed by a survey was performed to investigate whether the model could assist software practitioners in writing their user stories in a simulated environment. The study concludes that text prediction sped up the writing of the sentences and contributed to a more consistent and uniform set of user stories; this study resulted in a journal paper submitted to *Requirements Engineering Journal* and has been going through pair review since January 2025.
- The **Conclusion** wraps up this thesis by emphasizing the key research contributions, addressing its limitations, and outlining potential directions for future work.

CHAPTER I

BACKGROUND

In a broader aspect, this research uses artificial intelligence to support software requirements specification, most precisely user stories in ASD projects. Therefore, the background is divided into two broader topics: *software requirements engineering* and *natural language generation and metrics*. This chapter aims to create background knowledge for the reader to comprehend the achievements of this research and discover ways to adapt it to the reality of software practice.

It begins by presenting the RE theory and its application to SE in agile environments. In addition, we narrowed our study to requirements specification, bringing empirical works demonstrating the problems software practitioners face when writing requirements statements. It gives us the base knowledge to present user stories and how they are employed in the industry. To finish this first part, we discuss software requirements quality and the diverse approaches available to evaluate user stories. We present the Automatic Quality User Story Artisan (AQUSA) tool used as a benchmark in this thesis to assess the quality of user stories written with support from text prediction systems.

Secondly, we present the AI techniques employed in this thesis to support the writing of user stories. We begin by presenting text prediction systems and their statistical methods. We also stage the deep learning algorithms that can generate text automatically. To close, we discuss the metrics used to evaluate the text quality generated.

1.1 REQUIREMENTS ENGINEERING

The theory described by RE is used to build any product. The industry relies on RE tasks to produce cars, computers, televisions, airplanes, etc. When the companies have a proper engineering specification, it facilitates the process of writing user's manuals, patenting and selling the product (Schneider & Berenbach, 2013).

Each industry domain needs to organize requirements descriptions, though several default tasks can be reused across the industry. Thus, the Requirements Engineering Board (IREB)¹ provides the Certified Professional for Requirements Engineering (CPRE) certification scheme to prepare professionals to work on requirements engineering. These professionals are ready to ensure the quality of final products by using the theory behind RE.

In the context of Software Engineering (SE), RE is a step of the software development process (Figure 1.1). This task is responsible for identifying, organizing and tracking the client's needs during the software development process (Aurum & Wohlin, 2005). It is executed in the first phases of the project to support the writing of the project scope, identify stakeholders and spot the needs of software end users.

According to Wieggers & Beatty (2013), software requirements engineering can be divided into two phases: development and management (Figure 1.2). In the first one, four phases are executed to discover, document and inspect software requirements.

- **Elicitation:** This stage involves all tasks related to requirements discovery. Development teams can combine diverse techniques to gather requirements from stakeholders and users, such as interviews, prototyping, document analysis and others.

¹<https://www.ireb.org/en>

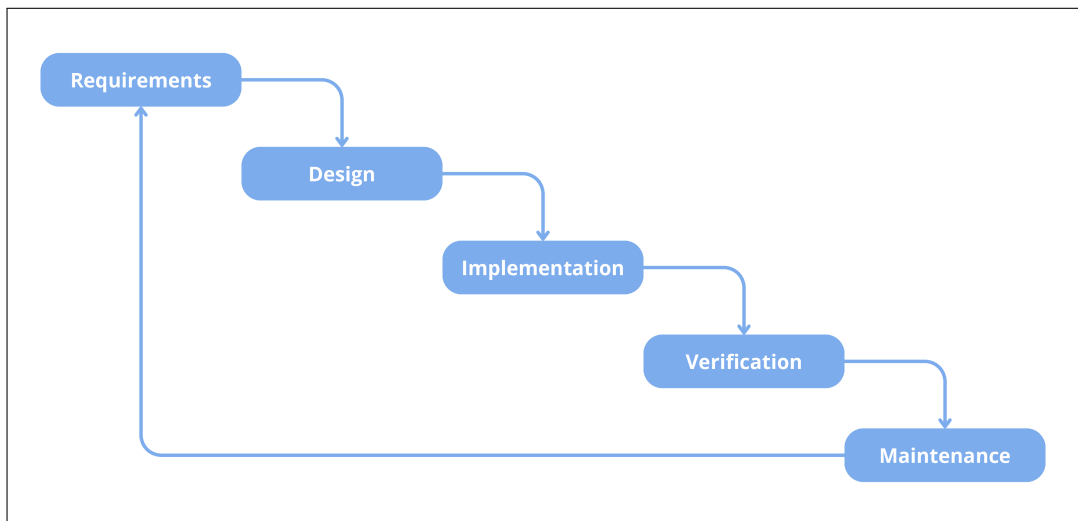


Figure 1.1 : Waterfall model for software development. The waterfall model is not ideal when teams can communicate informally and software requirements change frequently. In such cases, iterative development and Agile methods are more suitable (adapted from Sommerville (2016)).

- **Analysis:** During this step, the development team and stakeholders work together to better understand the requirements by identifying gaps, negotiating priorities and breaking down high-level requirements in better detail.
- **Specification:** It consists of persisting the requirements collected in a standard format that all stakeholders can agree upon and have access to whenever necessary.
- **Validation:** In this last stage, the development team and stakeholders confirm their requirements to build the right software product.

During the requirements management phase, all the necessary changes are managed to keep the project on track with the customer’s needs. Requirements can be added, updated or removed, and their dependencies and relationships must be identified and controlled to minimize the disruptive impact on the project (Wiegiers & Beatty, 2013). Aurum & Wohlin (2005) also divide the requirements management into five practices.

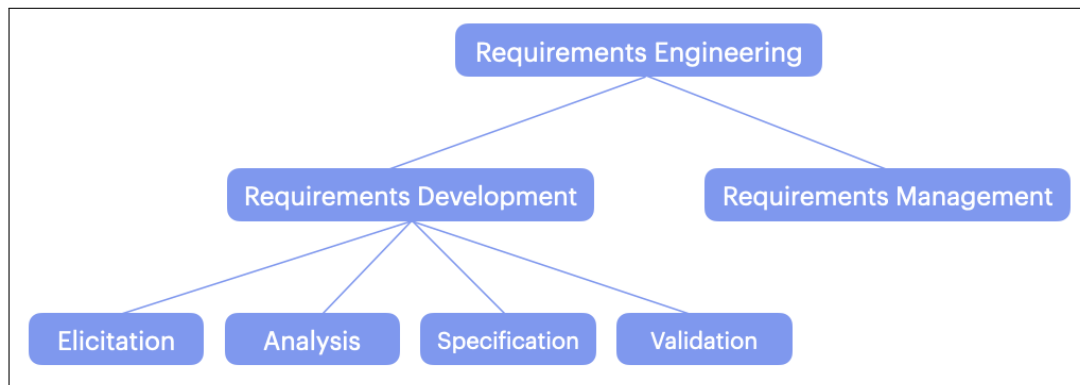


Figure 1.2 : Subdisciplines of software requirements engineering - adapted from [Wieggers & Beatty \(2013\)](#).

- **Requirements Elicitation, Specification and Modelling:** This iterative work defines software requirements. It is repeated throughout development to ensure the right software product will be built.
- **Prioritization:** Requirements changes also require re-ranking the requirements according to the market and stakeholders' needs.
- **Requirements Dependencies and Impact Analysis:** When requirements change, they usually affect the software product and the business. This step helps development teams and stakeholders track the impact of the changes.
- **Requirements Negotiation:** Development teams cannot take the requirements decisions alone. Different points of view need to be considered to make the right decision.
- **Quality Assurance:** Development teams and stakeholders shall work together to ensure the quality of the requirements defined, as it can affect the whole software development process.

1.1.1 REQUIREMENTS IN AGILE SOFTWARE DEVELOPMENT

In agile projects, the level of documentation varies depending on the nature of the product software under consideration and the contract between the development team and stakeholders. Generally, lightweight documentation that is mutually acceptable tends to be more effective (Fernández *et al.*, 2017). This behaviour responds to the agile manifesto, where working software is more valuable than comprehensive documentation.

The essence of ASD relies on iteration and continuous feedback from stakeholders to deliver the right software product. Unlike traditional sequential phase approaches, ASD supports the frequent changes required in dynamic development environments, offering better flexibility (Figure 1.3). In this context, requirements are carried out iteratively during all the stages of the development process, allowing teams to adapt to changes more quickly. These iterative and incremental approaches reduce waste and rework because the teams gain feedback from stakeholders (Institute, 2017).

To facilitate this iterative work, diverse approaches can be combined to gather and document requirements effectively. ASD generally emphasizes minimal documentation, favouring concise and practical artifacts such as user stories, product backlogs, prototypes, and use case scenarios (Schön *et al.*, 2017). These artifacts help development teams to keep the focus on the user while avoiding lengthy and complex documentation (Inayat *et al.*, 2015).

Recent studies tackling the aspects of RE in ASD confirm that user stories are the most used artifact for representing software requirements in ASD projects (Schön *et al.*, 2017; Inayat *et al.*, 2015; Fernández *et al.*, 2017). They are easy to employ and maintain and serve as reminders for future conversations among team members. The use story approach is better presented in Section 1.1.1.

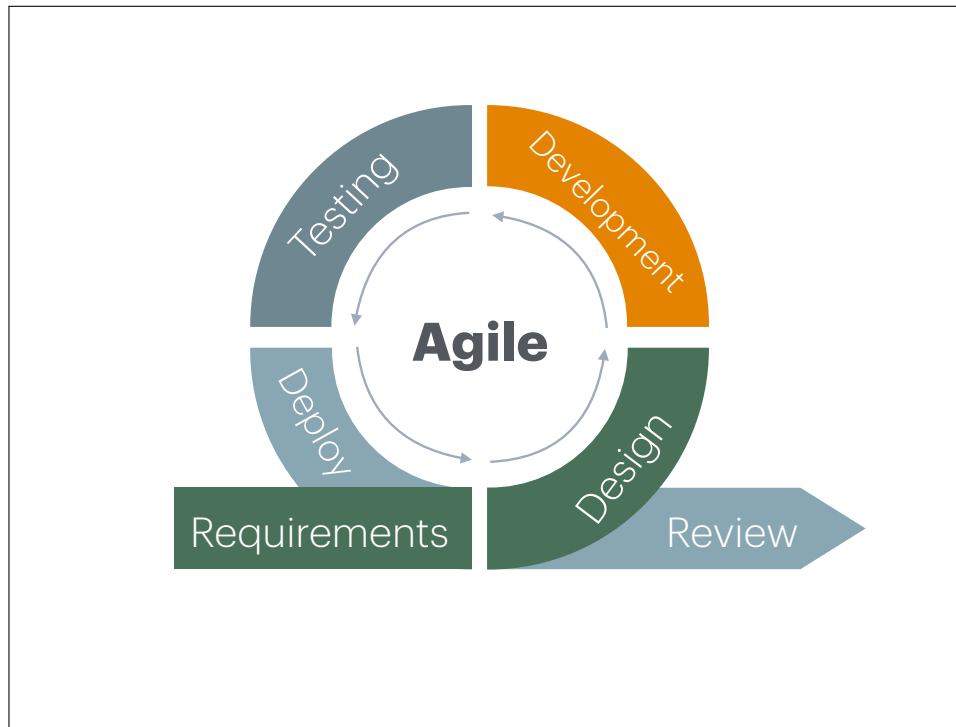


Figure 1.3 : Agile development process iteration (adapted from Pham (2018)).

In addition, prototypes are a powerful tool in ASD, acting as a shortcut to help stakeholders validate software requirements without spending time on software infrastructure and coding (Schön *et al.*, 2017). Paper prototypes are commonly utilized in software development. They consist of drawing user flows to show an ideal user navigation and to decide the functionality that the user needs; wireframes presented in a testable form (paper or clickable); coded prototypes, such as those using HTML navigation; and visually designed, high-fidelity prototypes (McElroy, 2017).

Prototyping aligns with the agile philosophy of experimentation and iterative development. Every piece of software delivered during the development process can be seen as a prototype, allowing one to validate requirements and receive stakeholder feedback. It helps cultivate a mindset focused on continuous improvement and ongoing feedback, significantly enhancing the final software product (McElroy, 2017).

Also, use case scenarios are a valuable tool in ASD as they describe the interaction of users with the system from a more technical viewpoint, providing a more detailed understanding and documentation of user stories. Ivar Jacobson introduced use case scenarios as part of the Object-Oriented Software Engineering (OOSE) methodology in the early 1990s. While they were not initially part of agile methodologies, use case scenarios have been adapted and integrated into agile software development practices (Jacobson, 1992). A use case scenario comprises some standard elements; others can be included or removed, such as use case name, actors, pre and postconditions, primary, alternative and exception flow, extensions, and assumptions. Table 1.1 presents a use-case scenario example. Note that a user story was used to describe the use case. It reinforces the cross-use of the artifacts to create a better understanding of the software system. The use case also cites the business rules involved in the process.

It is worth mentioning that ASD requirements artifacts can be combined to understand better the system to be developed. Modern approaches support the elicitation and specification requirements in ASD projects. For example, the book *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days* (Knapp et al., 2016) presents an iterative and fast approach involving user interviews, storytelling, prototyping and collaborative work that can be applied on software engineering to describe the expected software behaviour. The entire process can be employed in one week of work and aims at helping stakeholders and development teams validate products and learn more about market and user needs.

At this point, it is possible to conclude that modern approaches to validate software products are taking place in ASD projects and supporting a more holistic and iterative management of software requirements. How software requirements are prepared and validated nowadays differs from traditional phased requirements engineering. The agile mindset en-

Table 1.1 : Use case scenario example extracted from a project of system integration of a transportation company (author’s personal archive).

Use case ID	US01
Title	Send delivery data to transportation company
Use Case Description	AS a <Client> I WANT to share delivery data SO THAT they can route my deliveries.
Actors	<Client>
Precondition	PRE01: Must have login and password to access the API PRE02: Must have permission to send delivery data using the API
Post-condition	POST01: Must store the tracking ID generated by the API after concluding the operation. POST02: Must save a copy of the data sent.
Main path	<ol style="list-style-type: none"> 1. The actor opens its preferred REST API client. (CM01) 2. The actor selects the POST operation and informs the endpoint URL for sending data delivery to us. 3. The actor copies and paste the JSON data to the body of the message and clicks on the OK button. 4. The REST client presents a message asking for credentials to finish the operation. 5. The actor types the login and password and clicks on the OK button. (EP01, EP02, EP03, EP04, EP05) 6. The REST client processes the data and returns a response.
EP01: Exception path	6. Bad credentials (ERROR 401 - Unauthorized).
EP02: Exception path	7. The actor contacts the IT department to get new credentials (step 3).
EP03: Exception path	6. Wrong URL (ERROR 404 - Not found).
EP04: Exception path	7. The actor updates the URL address and tries again (step 3).
EP05: Exception path	6. Incorrect format (ERROR 422)
Business Rules	7. The actor updates the data body field and tries again (step 3).
Comments	6. Incorrect syntax (ERROR 400) 7. The actor updates the data body field and tries again (step 3). 6. Route not found (ERROR 402) 7. The actor updates the data body field and tries again (step 3). BR007 CM01: The <Client>can use any other way to interact with our API.

courages iterative and incremental work on requirements and brings more opportunities for changes during project development. The outcome is lightweight documentation supported by communication among team development members.

USER STORIES

Beck & Andres (2004), the creators of the Extreme Programming process, proposed the concept of story cards. The cards were written using short sentences to describe the system’s behaviour as a goal to incentivize the team discussion. All the cards were split into tasks to be completed by the development team. Later, Cohn (2004) refined the definition of story cards. He stated that story cards contain a short description of the user or customer-valued

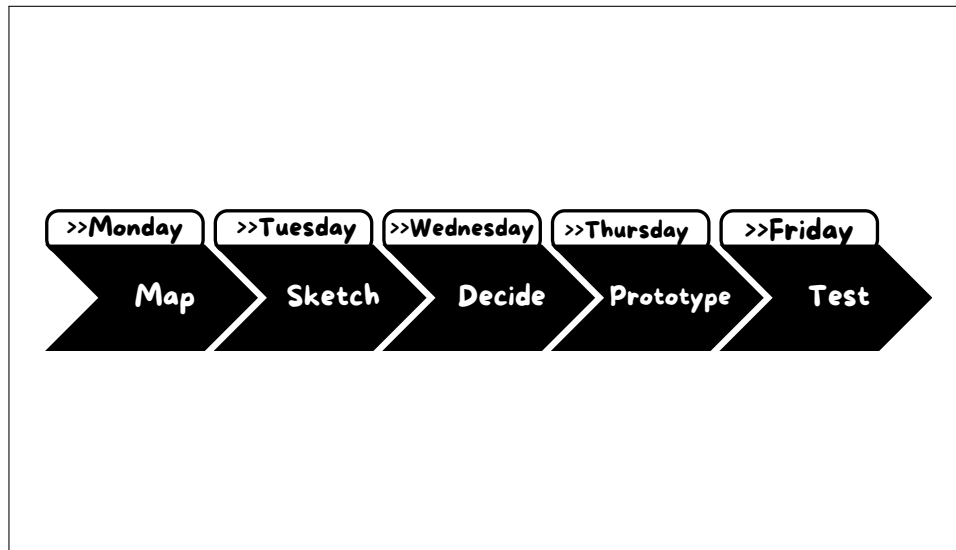


Figure 1.4 : Phases of the Spring method to validate new software products. On Monday, the participants map out the problem and choose a target to keep the focus. On Tuesday, they brainstorm and sketch new and existing ideas on paper. On Wednesday, some hypotheses are set to test the ideas created the day before. On Thursday, the participants create realistic prototypes, and on Friday, they test the prototypes with target customers. [Knapp et al. \(2016\)](#).

functionality, but the essential parts are the conversations between stakeholders and developers about the story. He also described guidelines for writing good stories to guide the software development process correctly.

Any team member or stakeholder can author user stories at various project stages. They typically emerge from conversations between team members and stakeholders or from various requirements sources such as project documents, standards, business rules, or emails ([Cohn, 2004](#)). These user stories follow a semi-structured template to express valuable user functionality, and employing this template enhances productivity and the quality of deliverables ([Dalpiaz et al., 2019](#)). Each user story aims to define an *actor*, an *goal*, and the *benefit*, or business value delivered by that action. For instance, the actor (or role) could be the end-user, and a goal could be an operation made by the actor at the system or business level, such as

printing a document, scanning a barcode or returning a product to the warehouse. Finally, the benefit represents what the actor will accomplish through that goal.

As a <role>, I want <goal>, so that <benefit>.

It brings the practice of focusing on the client's perspective about the system and the business. The user story template is simple, so the writer cannot add too many details. So, rather than writing all details as stories, the better approach is to encourage communication between the development team and the customer. There is no problem in writing down some notes on the story cards, but the conversation is the goal when working with user stories (Cohn, 2004). The communication motivated by the user stories can improve the clarity and understanding of the software requirements. This is an example of a user story extracted and adapted from a project² of an online platform to support waste recycling:

As a user, I want to be able to get a list of nearby recycling facilities, so that I can determine which ones I should consider.

Lucassen *et al.* (2016b) evaluated the use and effectiveness of user stories in practice. They state that the user story is the minor representation of requirements software developers use to build new features daily. Using a template to express user stories is a standard industry practice. Different templates can be used to assist development teams in writing user stories. The most used user story template is *Connextra* (example above), which originated with agile coach *Rachel Davies* in the early 2000s (Cohn, 2019); From the survey performed by

²<https://data.mendeley.com/datasets/7zvk8zsd8y/1>

Lucassen *et al.* (2016b), only 15% of respondents do not use user stories templates. Those who use templates agree that the user story representation supported by templates contributes to productivity and work deliverable quality (Lucassen *et al.*, 2016b; Dalpiaz *et al.*, 2019).

User stories are essential in helping development teams build the right product (Lucassen *et al.*, 2016b). As they are smaller pieces of requirements, the stakeholders are pushed to depict requirements with more details to development teams. The more information they have, the easier it is to code the right product. It also reinforces communication and brings the development team and stakeholders to the stage in the early phases of the project. In fact, it has been shown that stakeholders like to work with user stories (Lucassen *et al.*, 2016b). This practice helps development teams to prevent later defects in the software. Recent studies confirm that user stories are the most used artifact for representing software requirements in ASD projects (Schön *et al.*, 2017; Inayat *et al.*, 2015; Fernández *et al.*, 2017).

On the other hand, (Pokharel & Vaidya, 2020) produced another study of user stories in practice where they found that not all agile practitioners use user stories. As shown previously, other approaches can also gather and specify requirements for ASD. Only 45% of their respondents are committed to applying user stories in agile software development, and only 15% have sound knowledge about this practice. In addition, they state that backend developers are more prone to not use user stories, as opposed to full-stack developers who take more advantage of this practice. They conclude that development experience alone is insufficient to gather client needs. Practical training on user stories and agile principles is required to play agile practices in organizations.

1.1.2 REQUIREMENTS SPECIFICATION IN PRACTICE

The increasing complexity of software has shifted RE from a preliminary task in SE to a critical focus throughout the software development process. The abovementioned theory is necessary for the software industry to apply RE activities in their projects. However, they are broader tasks with diverse application methods in the development life cycle. It is reasonable to state that RE is a complex problem-solving activity that involves making numerous decisions when dealing with heterogeneous environments and distributed work (Aurum & Wohlin, 2005).

More empirical work is necessary to study the practice of RE in the industry and contribute by developing new methods and tools. The focus of this research is on requirements specification. Thus, we investigated what empirical studies have been concluded about the practice of requirements specification in the industry. Wagner *et al.* (2019) designed a survey destined for software development companies to define the *status quo* in requirements engineering. They found that most requirements are elicited via interviews with users and stakeholders and are specified using free-form and minimal textual constraints, working as reminders for further discussions. Also, the companies usually define acceptance criteria for their requirements as a way to comply with minimal quality guidelines.

This finding is aligned with Khan *et al.* (2021), who empirically evaluated the most critical RE practices on Global Software Development (GSD). It involves a market practice of outsourcing software development to vendor companies, often geographically distributed. One of the RE critical practices identified by the authors is aligned with the requirements specification stage in RE: the definition of standard templates for describing requirements. They state that it is a crucial practice among development teams because it improves the quality of the requirements statements by increasing consistency and decreasing incompleteness.

In a heterogeneous environment like in GSD, using requirements templates can facilitate communication among team members and stakeholders.

With a similar goal, [Fernández et al. \(2017\)](#) prepared a survey to investigate the RE pain points in software companies. Regarding requirements specification, they found that *incomplete and/or hidden requirements* is one of the most cited problems in RE and mostly related to project failure. In general, the leading reported causes linked to this problem are *team weak qualification* and *lack of experience of the RE team members*. The report finishes by giving some first countermeasures to help companies prevent this issue. One of them is to improve their completeness checking, respecting their development method.

The value of empirical research in RE is noteworthy in discovering how software practitioners apply RE in the context of SE. The studies above concluded that development teams use different formats to specify requirements, from free-form styles to templates with minimal textual constraints. Still, they need more qualifications in RE and usually lack the experience to ensure the quality of software requirements written.

In this way, some authors have been working to automate tasks involved with requirements to support development teams in gathering, specifying and managing the RE life cycle. [Umar & Lano \(2024\)](#) published a systematic review of RE automation inspecting the approaches proposed. They grouped analysis and specification into one category as proposed by [Sommerville \(2011\)](#) and found that this is the most automated phase in RE, with requirements validation as the least automated one. The UML class diagram is the most widely generated model, probably because it gives a high-level definition of the requirements and the foundation for the software design phase.

Natural Language Processing (NLP) techniques are the most used to develop automated tools (mainly part-of-speech (POS) tagging and parsers ([Umar & Lano, 2024](#))), followed by

machine learning algorithms and ontologies. Among Machine Learning (ML) algorithms, researchers mostly used the following ML algorithms in RE activities: *Decision Tree*, *Support Vector Machine*, *Naïve Bayes*, *K-nearest neighbour Classifier*, and *Random Forest* (Khan, 2024). To assess the performance and reliability of these automated solutions, controlled experimental methods remain the evaluation approach most frequently employed (Umar & Lano, 2024).

Zhao *et al.* (2021) investigated how researchers have been employing NLP in RE, or *NLP4RE* for short. Initially, they found that most researchers are more concerned about delivering solution proposals than investigating and evaluating problems in the area. Also, they work to assess empirically their proposal in labs instead of doing a field study. According to the authors, analysis is the most researched phase of RE, confirmed later by the findings of Umar & Lano (2024), followed by management, elicitation, and modelling. To close, the authors recommend researchers expand the studies to other RE phases, mainly those that overlap with software development in design and testing tasks. Empirically, other evaluation methodologies could be used to assess the quality of the tools proposed, ensuring a better tool evaluation in real scenarios of use.

Most recently, Kokol (2024) mapped the relevant studies intersecting AI and SE. They also highlighted works applying NLP techniques in RE phases supporting tasks such as domain concept identification, requirements classification, different levels of validation, and requirements elicitation. The paper indicates that the AI models need more training aligned to the domain where they will be employed, ensuring more accurate results.

The SLR presented in automation for RE points in the same direction: most of the research produced in RE is related to the analysis phase, usually followed by the specification phase. Also, diverse authors agree that the proposed tools and frameworks lack more empirical

evaluation, leading to ineffective solutions in the industrial environment. NLP techniques are adopted mainly among researchers for text preparation or tool production. To finish, authors could be more committed to depositing their tools publically to ease the validation and possible industry adoption.

1.2 REQUIREMENTS QUALITY

In modern SE, quality management relies on establishing a quality culture among development teams. Every developer is responsible for software quality, and synchronized actions are set to ensure that quality is maintained (Sommerville, 2016). Thus, testing efforts are employed earlier in the development process (Soeken *et al.*, 2012). The testing step needs to be based on user requirements to discover any defects that may cause the system to fail to meet the client's requirements (Chopra, 2018). However, when the user needs are poorly written, they can communicate inaccurate messages to the development team, affecting the coding until the software delivery.

The validation phase of RE is essential to ensure the requirements statements are well-written and contain all the necessary information to build the solution that meets the business and user needs. In this phase, development teams can work together to review the requirements documents written and write acceptance criteria and tests (Wieggers & Beatty, 2013).

Based on this scenario, there are some ISO standards able to guide the requirements engineering task in software product development. These standards describe levels, processes, vocabulary, and also how requirement tools should be built to support development teams. Many of these standards cross-reference each other, and some are obsolete due to technological evolution. Schneider & Berenbach (2013) produced a valuable study correlating ISO standards of software requirements.

One specific ISO standard is fundamental knowledge for this research: ISO/IEC/IEEE 29148:2018 (ISO, 2018). This standard helps the reader to understand the components present in requirements statements and how to assess the quality of these descriptions. It was also relevant to guide subsequent studies about requirements quality evaluation.

ISO/IEC/IEEE 29148:2018 is a standard responsible for *describing processes for requirements engineering*, and also for setting specifications for requirements documentation. It begins by presenting terms and definitions related to the requirements domain that facilitates reading technical aspects. This standard describes the concepts behind requirements engineering and the processes used to manage requirements specifications. Also, this standard introduces the characteristics of individual requirements and their sets, language criteria, and quality aspects. This knowledge helps the reader understand the components present in requirements statements and how to assess the quality of these descriptions.

According to the standard, there are nine quality attributes of *individual requirements*.

- **Necessary:** The requirement defines an essential capability, characteristic, constraint and/or quality factor.
- **Appropriate:** The specific intent and amount of details of the requirement is appropriate to the level of the entity to which it refers.
- **Unambiguous:** The requirement is stated so that it can be interpreted only one way. The requirement is expressed and is easy to understand.
- **Complete:** the requirement sufficiently describes the necessary capability, characteristic, constraint or quality factor to meet the entity need without needing other information to understand the requirement.

- **Singular:** The requirement states a single capability, characteristic, constraint or quality factor.
- **Feasible:** The requirement can be realized within system constraints (e.g., cost, schedule, technical) with acceptable risk.
- **Verifiable:** The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exist.
- **Correct:** The requirement accurately represents the entity need from which it was transformed.
- **Conforming:** The individual items conform to an approved standard template and style for writing requirements, when applicable.

This ISO standard also contributes by defining the quality characteristics of a *set of requirements*.

- **Complete:** The requirements are comprehensive and independently outline the essential capabilities, characteristics, constraints, or quality factors to fulfill the entity's needs without requiring additional details.
- **Consistent:** The set of requirements includes distinct individual items, free of conflicts or overlaps with other requirements in the set, and uses consistent units and measurement systems. The terminology within the requirements is uniform, meaning the same terms are used consistently throughout the set to convey the same meanings.
- **Feasible:** The requirements can be executed considering the project constraints, such as time, cost and technical capabilities.

- **Comprehensible:** The set of requirements is crafted to clearly define what is expected by the entity and how it relates to the overall system it is part of.
- **Able to be validated:** It is feasible that fulfilling the requirement set will meet the entity's needs considering the project constraints.

1.2.1 USER STORIES QUALITY

Requirements quality is a comprehensive software engineering subarea that constantly draws researchers' attention. Indeed, faulty requirements can affect the subsequent software engineering process and lead to the delivery of low-quality products. User stories are not an exception in this case, as they are employed in the early stages of the software development life cycle to act as reminders for future refinement. Their straightforward structure helps development teams write concisely; however, studies point out that user stories fail to capture all the user requirements and are prone to multiple interpretations (Ordoñez *et al.*, 2015).

User stories do not mean to be a requirement document but act as a reminder for future refinement (Shore & Warden, 2021). Therefore, the guidelines for RE quality cannot be integrally applied to the stories (as the standards presented in section 1.2). Some authors proposed their methodologies to assess the quality of user story statements (Cohn, 2004; Heck & Zaidman, 2014; Lucassen *et al.*, 2015). These works are valuable because ambiguous user stories can affect the requirements refinement and software coding and deliver the wrong product to stakeholders and users (Amna & Poels, 2022a).

Initially, Cohn (2004) presented six quality attributes to evaluate user stories in ASD.

- **Independent:** User stories cannot be dependent as they lead to prioritization and planning problems. Independent user stories are more accessible to estimate and track along the project.
- **Negotiable:** User stories are not a contract nor a requirement to be developed. They are short descriptions acting as reminders for future refinement and negotiation.
- **Valuable to users or customers:** Some stories are not valuable for the user but only for the customer (who pays for the software). Usually, non-functional requirements bring more value to customers than to users. Every user story shall be written to bring value to users or customers.
- **Estimable:** Developers should be able to estimate the effort to work in a given user story. Developers shall have domain and technical knowledge to estimate user stories correctly.
- **Small:** User stories cannot be too big or too small. If so, you cannot use them in planning sessions. Some small stories can be combined as a way to reduce planning time. When stories are too big, they can be split into smaller ones for better tracking.
- **Testable:** Stories must be written to be testable. Without a testing approach, ensuring the story was successfully developed is impossible. Tests should be automated whenever possible to meet the high demand for changes in ASD environments.

The quality attributes for user stories presented by [Cohn \(2004\)](#) represent the base guideline for those who desire to employ user stories in ASD. The acronym INVEST ([Wake, 2003](#)) has been suggested by Bill Wake, author of *Extreme Programming Explored* ([Wake, 2002](#)).

Following [Cohn \(2004\)](#), [Heck & Zaidman \(2014\)](#) proposed the *Agile Requirements Verification Framework* to evaluate the quality of agile requirements (user stories and feature requests). The framework defines three macro quality criteria:

- **Completeness:** They consider three levels of detail: basic, required and optional. By doing that, they differentiate mandatory and optional elements to be present in the requirement statements.
- **Uniformity:** The format of the requirements statements shall follow a standard to minimize miscommunication and facilitate project management; the usage of support tools and templates is incentivized, and also the addition of extra comments should be managed to avoid standard conflicts.
- **Conformance:** The requirements shall be consistent and correct. It means they should attend to the INVEST attributes defined by [Cohn \(2004\)](#), use the proper language without contradictions, and specify the problem. If the language is domain-specific, a glossary should be included. Requirements statements should be unique, without duplication, and atomic, representing only one customer or user request at a time.

Based on the work produced by [Heck & Zaidman \(2014\)](#), [Lucassen et al. \(2016a\)](#) proposed the Quality User Story (QUS) framework. This framework defines 13 language criteria for user story quality divided into three categories: syntactic, semantic, and pragmatic. QUS also defines quality criteria for a set of user stories to verify the quality of a complete project specification. We rely on the QUS framework to evaluate the quality of the user stories generated by our text prediction model (better discussed in Chapter 4); Table 1.2 presents the 13 language criteria defined by the QUS framework.

Table 1.2 : 13 language criteria of the Quality User Story (QUS) framework (adapted from Lucassen *et al.* (2015)).

Criteria	Description
Syntactic	
- Atomic	A user story expresses a requirement for exactly one feature
- Minimal	A user story contains nothing more than role, means and ends
- Well-formed	A user story includes at least a role and a means
Semantic	
- Conflict-free	A user story should not be inconsistent with any other user story
- Conceptually sound	The means expresses a feature and the ends express a rationale, not something else
- Problem-oriented	A user story only specifies the problem, not the solution to it
- Unambiguous	A user story avoids terms or abstractions that may lead to multiple interpretations
Pragmatic	
- Complete	Implementing a set of user stories creates a feature-complete application, no steps are missing
- Explicit dependencies	Link all unavoidable, non-obvious dependencies on user stories
- Full-sentence	A user story is a well-formed full sentence
- Independent	The user story is self-contained, avoiding inherent dependencies on other user stories
- Scalable	User stories do not denote too coarse-grained requirements that are difficult to plan and prioritize
- Uniform	All user stories follow roughly the same template
- Unique	Every user story is unique, duplicates are avoided

Following the framework, the authors also developed the AQUASA tool to automatically evaluate the quality of user stories based on specific attributes of the QUS framework (Lucassen *et al.*, 2016a). This tool uses NLP algorithms to analyze the structure of individual and sets of user stories. It can automatically identify the language criteria defined by the framework and point out the defects that need user correction. This tool was coded in Python using the Flask microframework. It also relies on Stanford CoreNLP (Manning *et al.*, 2014) and the Natural Language ToolKit (NLTK) (Bird *et al.*, 2009) tools, using a set of algorithms based on language heuristics. It receives a list of user stories in text format as input, runs the linguistic analysis and outputs a report containing all user stories with the quality analysis result. Each user story can have neither or many issues according to the QUS framework classification. In the evaluation performed by the authors, the AQUASA tool detected the total amount of user stories with errors with 97.9% recall and 84.8% precision. It demonstrates how the AQUASA tool and QUS framework can be used in industry environments to support user story quality analysis.

Unfortunately, the AQUUSA tool cannot analyze all the quality attributes defined by the QUS framework. All the *syntactic* criteria are covered by the tool; only the following *pragmatic* aspects are considered: *explicit dependencies, uniform and unique*; neither *semantic* attribute is covered because it would require a deep understanding of the content of the requirements. Despite this, the tool has received attention from researchers as it points a pivotal direction for user story analysis and is a starting point for future research.

Although some studies proposed new guidelines to evaluate user story quality, the agile manifesto (Beck *et al.*, 2001) highlights that ASD teams are responsible for defining the standards that better work for them (Schwaber & Sutherland, 2020). Cohn (2009) emphasizes that team-defined standards enable agility and ownership, making teams more invested in following best practices. Therefore, user stories are not an exception to this concept: development teams can work collaboratively to define their quality standards, ensuring all team members follow the same method when writing and evaluating their user stories.

The quality attributes defined by Cohn (2004); Heck & Zaidman (2014) are general guidelines that ASD teams can use to create specific standards more adapted to their domains. Lucassen *et al.* (2015) took a step forward, breaking down Heck & Zaidman (2014) framework and defining the QUS framework with more strict attributes to evaluate user stories. The development of the AQUUSA tool makes it easier to apply the QUS framework to real scenarios of user story writing (even if it does not cover all the attributes defined yet); it is crucial to remember that the goal of the user stories is to incentivize conversation and not serve as a requirement document (Shore & Warden, 2021). In this way, it is acceptable that development teams have ownership of how to write their user stories to ensure that customer requirements are considered during refinement meetings.

To supplement our study about user story quality, we highlight the systematic review of user story ambiguity published by [Amna & Poels \(2022a\)](#). This work makes an essential contribution by defining and analyzing the different levels of ambiguity in user stories (i.e., lexical, syntactic, semantic, pragmatic). Their dataset brings 36 studies published from 2001 to 2020 that evaluate ambiguity in user stories. They found three main gaps in research on this topic. First, there is a need for more research on human behaviours and cognitive factors that cause ambiguity. Second, the studies found that a set of user stories is rarely considered when analyzing ambiguity. They only focus on unique statements, but a set of user stories is prone to inconsistency, a kind of ambiguity. Third, the solutions proposed do not consider ambiguity at different linguistic levels, as presented by the authors. The studies need to rely more on linguistic theory to work with ambiguity in user stories.

To conclude, user story quality is an open topic in user story research, with many new publications coming every year. The SLR prepared by ([Amna & Poels, 2022a](#)) summarizes a bit of the state-of-the-art in the topic. On one side, ASD states that development teams are responsible for defining quality attributes for user stories in their projects. On the other hand, it is known that lousy user stories can lead development teams not to consider all the stakeholders and user requirements during the planning stage. It raises questions about the maturity of the ASD teams in defining quality attributes and which methodologies they should use to base their choices. In this scenario, we chose the QUS framework and the AQUASA tool as evaluation methods for our research, as they offer a more strict guideline for user story evaluation through an automated tool.

Quality attributes considered in this research

In Chapter 4, we used the AQUASA tool to evaluate the user stories produced by the participants of our controlled experiment who have written the statements using one of our text prediction models.

1.3 TEXT PREDICTION

Text prediction systems estimate the following or most probable tokens to occur in a sequence based on a text typed by the user. Hamarashid *et al.* (2022); Garay-Vitoria & Abascal (2006) presented essential works by summarizing all techniques used to build text prediction models and the aspects involved in constructing these systems. Figure 1.5 depicts how the methods are correlated.

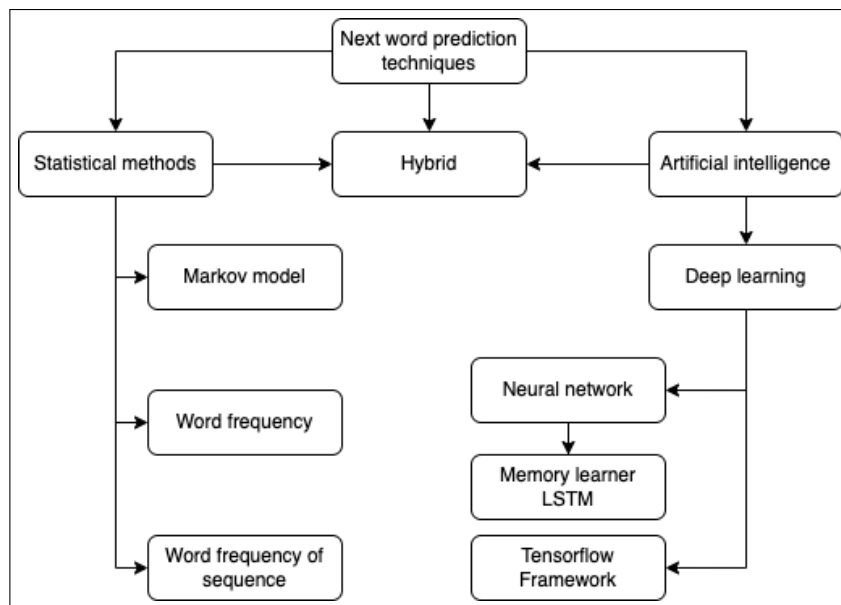


Figure 1.5 : Taxonomy of next word prediction systems Hamarashid *et al.* (2022).

Some authors can refer to text prediction as word prediction. For this thesis, we consider that they refer to the same concept, as a unique word can also be considered part of the text.

The ability to produce text besides unique words depends on the capacity of the trained model. More primitive models could make more straightforward text prediction, while most recent approaches [Buddana *et al.* \(2021\)](#); [Shini & Kumar \(2021\)](#); [Soam & Thakur \(2022\)](#) can build valuable text generation models able to work in different contexts generating n-grams.

These systems have become famous in assistive technologies and are a vital support to people with disabilities interacting with devices and other people. However, newer applications appeared, where these systems were used to facilitate the operation of devices by humans to perform daily tasks ([Hamarashid *et al.*, 2022](#)).

1.3.1 PROBABILISTIC METHODS

Text predictors are not novel in natural language processing. They consist of several different approaches that can be combined to predict words in sentences. This section presents the statistical methods and deep learning algorithms used.

WORD FREQUENCY

Calculating the frequency a word repeats through the text is the most straightforward way to estimate its relevance. This relevance can be seen as its probability of occurrence in the text. Thus, most relevant words have a higher probability of appearing again. In this approach, all word frequencies are calculated and ordered decreasingly. A prediction system can use this list to make predictions, prioritizing the words at the top of the list ([Garay-Vitoria & Abascal, 2006](#)). It can be expressed by the following formula ([Sparck Jones, 1972](#)):

$$\mathbf{tf}_t^{(c)} = \sum_{\forall d \in \mathcal{D}^{(c)}} tf_{t,d}$$

being $tf_{t,d}$ the number of word occurrence t in the document d that belongs to the set $\mathcal{D}^{(c)}$ of corpus documents c .

In this model, the system only predicts one word each time, not considering the context of the writing. This systems always present the same prediction suggestions for a particular sequence. As a result, most of the time, the suggestions might be inappropriate (Ghayoomi & Momtazi, 2009).

N-GRAM MODEL

The n-gram model is an extension of the word frequency model presented before. In this approach, the previous words are considered to predict other words. When only one word is used to make predictions, it is called a Bigram model (Hamarashid *et al.*, 2022). If two words are considered, then it is a Trigram model. Besides, it is an N-gram model that utilizes more than two words to process a prediction. Jurafsky & Martin (2009) work had the most significant influence on this approach. They presented valuable insights and best practices to develop functional N-gram models.

The frequency or probability of word sequence can be expressed as follows:

$$P(w_{1:n-1}) \approx \sum_{k=1}^n P(w_k|w_{k-1})$$

This is a usual approach for text prediction, as it is relatively simple to apply and does not present high computational complexity. This is a purely statistical method; it does not depend on grammar rules (Hamarashid *et al.*, 2022).

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0	0.021	0.0027	0.056	0
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Table 1.3 : Bigram probabilities for eight words extract from a corpus. [Jurafsky & Martin \(2009\)](#)

Table 1.3 presents the bigram probabilities extracted from the *Berkeley Restaurant Project* corpus of 9,332 sentences ([Jurafsky & Martin, 2009](#)). In this table, it is possible to see that the likelihood of "want" being the subsequent word of "I" is 33%. In addition, the probability of "to" occur after "want" is 66%. They are examples of high probability where the model will predict those subsequent words. Lower probabilities will be ordered decreasingly or even declassified. This table can be extended to present the probabilities to N-gram sentences.

MARKOV MODEL

A Markov Model is a stochastic model used to model randomly changing systems, predicting that future states rely on the current ones and not on past events. Markov model is a generalization of the N-gram model. It can be applied to other things than words or texts ([Taylor, 2020](#)). Its goal is to model the state of a system with a random variable that alters over time. In other words, the next predicted word is based on the probability of a word in the text corpus ([Hamarashid et al., 2022](#)). The formula of the Markov chain is written as shown below:

$$P(W|W_{t-1}, W, \dots W_n) \approx P(W_1|W_{t-1})$$

Markov Models are easy to implement and do not demand high computational processing. However, this model does not have the memory to decide for a long-range ([Hamarashid *et al.*, 2022](#)). It can predict the next word, only having the last one as a parameter. Taking as an example the following two sentences:

1. I love cats.
2. I love dogs.

The unique words in the system are "I", "love", "cats" and "dogs". If the user starts writing a new sentence and so types the word "I", the probability that the system suggests the word "love" is 100%. Consequently, the likelihood of the word "cats" being told after the word "love" is 50%, or the same as "dogs," as both are the next word of "love" in the past sentences.

1.3.2 DEEP LEARNING

Deep learning is a subfield of machine learning. Both approaches come up with the same principle: predicting output from input. However, they do it differently, so deep learning has been categorized as a separate approach. They use artificial neural networks to inspire the human brain's learning process. Deep learning methods are distinguished by their ability to automatically learn data representations, which traditionally requires manual effort in machine learning. While deep learning can operate through supervised learning, its capacity to optimize features independently contributes significantly to its success. In contrast, traditional machine learning models require human intervention to design features and verify predictions, limiting

their level of autonomy. Even so, the main difference between the two approaches is the data required for training. Deep learning usually needs more data and, consequently, more computing power to deal with large amounts of data (Bokka *et al.*, 2019).

The deep learning approach relies on neural network algorithms to process data similarly to the human brain. These networks are a type of mathematical formalization that abstracts the connections among human neuron cells. Each neuron component of this system can dynamically control the *weights* in the decision-making process throughout its life. All this process is shared among the neurons as a goal to reach some goal — for example, text categorization and image recognition, among others (Lane *et al.*, 2019).

The adoption of neural nets to solve problems in the NLP field has improved the accuracy of prediction models. Three types of neural net algorithms are usually applied in NLP.

The **Recurrent Neural Networks (RNN)** has an internal memory that makes it easier to work with sequence modelling tasks. The current neuron output does not depend only on the input data but also on the previous outputs generated. It enables this algorithm to work with language generation, translation, sentiment analysis, etc.

There are studies in RE using RNNs to perform requirements classification (Kaur & Kaur, 2024; Gnanasekaran *et al.*, 2021). This practice saves time for development teams during refinement meetings and eases prioritization. Others (Kang *et al.*, 2019; Madala *et al.*, 2017) support the requirement-gathering process in extracting features in project repositories. This use can reveal hidden requirements and collaborate for a more complete final software product.

RNN are also employed in other areas, as an example of Rakib *et al.* (2019), who used RNN to build a text generation model for the *Bangla* language. This approach is corpus-based

trained, and the results outperform other methods, such as LSTM and Naïve Bayes with Latent Semantic Analysis; [Shini & Kumar \(2021\)](#) provided various techniques and different datasets that may be used to generate text summaries automatically. This work guides researchers or AI practitioners who desire to use RNNs to build models for summary generation.

Long Short Term Memory Networks (LSTM) inherits the same architecture as RNN but replaces the hidden layer with an LSTM unit. It has a more dynamic memory control, allowing the network to remember only the valuable information for the next prediction steps. All unnecessary data is removed from the system. To reach it, this network uses memory units (or *cells*) to combine previous states with current input. These cells decide what to keep in memory and what to dispose of. In NLP, LSTM can determine which words can be considered part of the context or not ([Lane et al., 2019](#)).

[Buddana et al. \(2021\)](#) compares the usage of RNN and LSTM to build text generation models. LSTM algorithm is a more robust solution for dealing with sequential and text data than RNN. Recurrent networks have their limitations in keeping track of long-term dependencies.

Transformer Neural Networks (TNN) utilize the transformer architecture to handle sequential data efficiently, making it particularly suitable for language-related tasks ([Vaswani et al., 2023](#)). Several variants of TNN are used for NLP tasks employing different encoders and decoders ([Tay et al., 2022](#)). Transformers have revolutionized natural language processing since their discovery in 2017, enabling *Large Language Models* to process and generate accurate human-like text.

LARGE LANGUAGE MODELS

The Large Language Models (LLM) are primarily built on TNN. They undergo training by processing extensive quantities of unlabeled text using a range of pretraining objectives. This process equips the model with the ability to autonomously learn from the text (Webber, 2023) generating its learning target (self-supervised learning). In a nutshell, a LLM can be used to create text from the input of a random text. Much of what is known about large language models was introduced and formalized by two keystone papers: Bidirectional Encoder Representations from Transformers (BERT) (Devlin *et al.*, 2019) and Generative Pre-trained Transformer (GPT) (Radford *et al.*, 2018). Both are trained in different ways and can also generate text differently.

Since then, numerous new models have emerged (Min *et al.*, 2024) to tackle text generation, offering various parameters and approaches. Many of these models allow the user to fine-tune using specific corpora and teach the model to generate more context-sensitive answers. GPT enables it to draft documents, write computer code, translate languages, analyze texts, and more. For this research, we mainly employed N-gram models. However, we also fine-tuned a version of GPT (Radford *et al.*, 2018) to generate user stories since it is one of the most popular LLM.

LLM have had a significant impact on several domains, including SE. For example, new studies are emerging to understand how LLM can support requirements elicitation and specification. Using LLM in this area is advantageous because it can improve brainstorming and idea exploration, save costs, and increase requirements accuracy and overall quality, enhancing productivity (Marques *et al.*, 2024).

However, there is a necessary discussion about the cons of using LLM in RE tasks. Some software specialists highlight the problems and limitations of using these models in RE, such as context bias, hallucination, reasoning errors, data privacy and ethical considerations. As RE is executed in the early stages of the software development cycle, errors created earlier can affect the whole software production and become more expensive to solve later (Borg, 2024). LLM needs more scientific evaluation as it is considered in new areas quickly. Still, in the meantime, the software industry has been taking advantage of using LLM in different steps of the development life-cycle.

Another critical point of discussion is whether RE professionals are ready to use LLM in their projects. Adopting LLM might demand precise prompts to give the proper output. Software practitioners should be trained to manipulate the models and reach the expected results correctly. At the same time, would the LLM be sensitive to the project context? Maybe some models could be fine-tuned or trained to be more accurate to the context, or the participation of experts should be constant to validate the outputs generated by the models (Arora *et al.*, 2024).

Adopting LLM in RE remains early. However, some research has pointed out the direction (Marques *et al.*, 2024; Borg, 2024; Arora *et al.*, 2024), and other empirical works (Ronanki *et al.*, 2023a; Zhang *et al.*, 2023) have shown valuable results by evaluating these models in project scenarios.

1.3.3 QUANTITATIVE METRICS

Natural language generation has been a longstanding goal for AI researchers, and it recently saw a significant breakthrough in the past decade, primarily due to the resurgence of deep neural networks. At the same time, we can finally generate consistent and human-like

texts, albeit imperfect. The evaluation metrics are still relatively simple (Table 1.4). For instance, one widely used metric is Bilingual Evaluation Understudy (BLEU) (Papineni *et al.*, 2002). BLEU quantifies the similarity between generated and reference texts by counting overlapping N-grams. It assigns a score from 0 to 1, with higher scores denoting closer similarity to the reference. BLEU’s enduring popularity is due to its simplicity and low computational cost.

Metric	Purpose	Focus	Use Cases
BLEU	Measures the precision of n-grams (word sequences) in generated text compared to reference text.	N-gram overlap between generated and reference text.	Machine translation, text generation, summarization.
ROUGE	Measures recall of n-grams and other text features between generated and reference text.	Recall of n-grams, word sequences, and word overlaps.	Text summarization, abstractive generation, content recommendation.
BERTScore	Measures the similarity between generated text and reference text using contextual embeddings from a pre-trained BERT model.	Semantic similarity between text using BERT’s embeddings.	Text generation, dialogue systems, question answering, summarization.

Table 1.4 : Summary explanation of the text prediction metrics used in this research.

A second metric, with numerous variations, is Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004). ROUGE is primarily designed to evaluate the quality of text generated by measuring the overlap of N-grams between the generated and reference texts, focusing on recall rather than precision. Furthermore, ROUGE is more suitable for tasks with multiple reference summaries. Compared to BLEU, ROUGE places greater importance on the comprehensiveness of the generated content. Nevertheless, ROUGE suffers from the same problem as BLEU: it cannot measure semantical similarities and is hence limited to syntactical matches.

A particularly noteworthy metric to consider is BERTScore (Zhang* *et al.*, 2020), which assesses sentence similarity by calculating the sum of cosine similarities between the embeddings of the tokens in two sentences. BERTScore is valuable as it effectively tackles two prevalent issues often encountered with N-gram-based metrics. The first one is the semantic comparison. For instance, BLEU & ROUGE fail to recognize that the sentence ["The building is well assembled"] is more similar to ["The house is nicely constructed"] than ["The house of horror"], but BERTScore would. Furthermore, BERTScore avoids the limitations of previous metrics when capturing distant relationships. A straightforward example is causality, such as A causes B vs B causes A Zhang* *et al.* (2020), especially in a long sequence.

1.4 CONCLUSION

This chapter introduced the fundamental knowledge needed to comprehend the remaining part of this thesis. We explored Requirements Engineering (RE) and its steps and how it is performed in Agile Software Development (ASD). We narrowed our focus to user stories as they are the artifacts target of this study. We also presented some empirical works about requirements specification. To conclude the session about RE, we discussed requirements quality more broadly and then narrowed the analysis to quality in user stories.

We also explored the topic of text prediction, presenting different probabilistic methods to generate text, including Machine Learning (ML) algorithms and Large Language Models (LLMs). Finally, we showed the quantitative metrics that can evaluate text automatically generated.

We state that using text generation models to write user stories is a promising approach that can significantly enhance the efficiency and quality of requirements engineering in ASD. This thesis does not aim to find the best text prediction model for user stories. Instead, it

evaluates the potential benefits of this approach through a controlled experiment with software practitioner participants. Hence, we performed sequential steps to build a text generation model and evaluated it using strict quality guidelines for user stories.

In Chapter 3, we used the N-gram approach to generate text for user stories and then compared the results with user stories generated by a fine-tuned GPT-3 model, using the BLEU, ROUGE, and BERTScore metrics for evaluation.

In Chapter 4, we enhanced the N-gram model and integrated it into a prototype tool for a controlled experiment. This experiment aimed to assess how text prediction could improve the quality of user stories. To evaluate the models's effectiveness, we used the AQUASA tool as a benchmark for comparison.

CHAPTER II

A COMPREHENSIVE SYSTEMATIC LITERATURE REVIEW ABOUT AUTOMATIC USER STORY GENERATION

2.1 INTRODUCTION

To the best of our knowledge, no prior secondary studies have specifically focused on automatically generating user stories. For this reason, we prepared a SLR to identify the state-of-the-art in the topic. We published this work on the *International Journal of Data Science and Analytics* in June 2024, entitled *Automatic user story generation: a comprehensive systematic literature review* ([DOI 10.1007/s41060-024-00567-0](https://doi.org/10.1007/s41060-024-00567-0)).

The remainder of the chapter presents this study's findings. We start by presenting similar review studies linked to the topic. Second, we present the research design employed to gather the dataset of papers. We then present the results, and the discussion points to the direction for future work. To close, we present the threats to validity and conclusion.

2.1.1 SIMILAR REVIEWS

We found some relevant secondary studies on user story practice to guide our research. These studies, detailed below, offer valuable insights.

[Amna & Poels \(2022b\)](#) conducted a systematic literature mapping of user story research. Their dataset comprised 186 unique peer-reviewed papers published in 2001-2021. They found that research on user stories practice is generally recent (2015-2021), representing 78% of the total of papers selected. Next, they noted that about 80% of the documents are solution-oriented, the other 20% being related to problem investigation or observation. Most proposed

solutions are based on algorithms to solve well-defined problems related to system design issues during requirements analysis and negotiation activities, i.e., algorithms to estimate project variables, prioritization, and project schedule optimization. For less well-defined problems, such as ambiguity in user stories, they (Amna & Poels, 2022b) found that the proposed solutions involve conceptual models or software models based on different artifacts to support the user story writing. While the models help to organize the relationship and dependencies between the stories, the artifacts, such as ontologies and glossaries, help the writers to produce better user stories. Otherwise, they conclude that there is a lack of tested solutions to mitigate the formulation of ambiguous user stories in RE.

Raharjana *et al.* (2021) presented a systematic review of NLP applied to user stories. Their dataset comprised 38 unique peer-reviewed papers published between 2009 and 2020. They found that the authors usually rely on Semantic Role Labelling (SRL) to identify the key abstractions of user stories (*who, what, and why*) and generate models/artifacts. Respectively, it contributes to a better understanding of the format of user stories and also accelerates the development lifecycle. Next, they mapped which NLP techniques are usually used in this context. They listed: *preprocessing, POS tag, named-entity recognition, vector space model, dependency, syntactic parse tree and bag-of-words (BoW)*. They also listed the challenges of using NLP in user story research: dataset heterogeneity, manual data tagging, context domain-dependent, human intervention and low recall and precision over the studies selected.

On a similar topic, Cheligeer *et al.* (2022) published a literature review on Machine Learning (ML) applied to requirements elicitation. RE or requirements discovery is the process of surfacing candidate requirements from different sources (Bourque & Fairley, 2014). Well-executed elicitation helps minimize software requirements incompleteness (Bourque & Fairley, 2014). Cheligeer *et al.* (2022) selected 86 articles for this study. They identified 15

different ML-based requirement elicitation tasks and 12 different data sources for building a data-driven model. In addition, they analyzed and categorized the techniques for constructing ML-based requirement elicitation methods into five parts. They found that *BoW language models* and *handcrafted features* are frequently used as a technique. However, an increasing trend towards using *embedding features* has also been observed. The most used ML algorithms to automatically elicit requirements are *Naive Bayes*, *Support Vector Machines*, *Decision Trees*, and *Neural Networks*. For evaluation, *Precision*, *Recall*, and *F1 score* are the most prevalent evaluation metrics applied to assess model performance. In the realm of natural language processing tools, *NLTK* (Bird *et al.*, 2009) and *CoreNLP* (Manning *et al.*, 2014) are widely recognized as the most commonly employed. In contrast, for ML training, *Weka* (Witten *et al.*, 2011) and *Scikit-learn* (Pedregosa *et al.*, 2011) are the most popular choices. The authors conclude that despite some studies lacking evaluation and concrete evidence, ML can theoretically and practically support requirements activities.

Also, Zhao *et al.* (2021) contributed to this area by mapping the state-of-the-art in natural language processing for requirements engineering (NLP4RE). In summary, they concluded that the past years were successful in providing technical solutions to cover the RE lifecycle, and now it is time to better evaluate those solutions with empirical assessment in real scenarios in order to lead to more powerful tools for RE. In addition, they found that although user stories, interview scripts, domain documents, use cases, and models are primarily used in ASD practice, they still need more attention from researchers.

Although the related literature studies presented in this section offered valuable insights into user story practices (Table 2.1), there were no studies on the automatic generation of user stories combining elicitation and specification processes for this goal. Integrating such research is paramount, as it will strengthen and provide clear guidance to ongoing research pursuits on this topic.

Table 2.1 : Summary of related studies.

Title	Goal	Concerns in research questions
User Stories and Natural Language Processing: A Systematic Literature Review (Raharjana <i>et al.</i> (2021))	Capture the current state-of-the-art of NLP research on user stories.	<ul style="list-style-type: none"> - Uses of NLP for user stories - Available NLP approaches for user stories - Challenges of using NLPs in user story
Systematic Literature Mapping of User Story Research (Amna & Poels (2022b))	Investigate user stories research to identify problems and solutions proposed by researchers, evaluate the level of maturity of the area and research gaps existent.	<ul style="list-style-type: none"> - Identify research areas related to user story technique - Diagnose the type of problems in the area - Analyze the kind of outcomes produced - Evaluate how the research is conducted - Identify the type of publications
Machine learning in requirements elicitation: a literature review (Cheligeer <i>et al.</i> (2022))	Summarizes and analyzes studies that incorporate ML and NLP into demand requirements elicitation.	<ul style="list-style-type: none"> - Analyze requirements elicitation activities supported by ML - Identify requirements data sources - Identify technologies, algorithms, and tools to build ML-based elicitation - Framework to build ML-based solutions for elicitation
Natural Language Processing for Requirements Engineering: A Systematic Mapping Study (Zhao <i>et al.</i> (2021))	Map the state-of-the-art in natural language processing applied on requirements engineering.	<ul style="list-style-type: none"> - Analyze publication data in NLP4RE - Analyze the state of the empirical research in NLP4RE - Identify the focus of research on NLP4RE - Inspect the tool development in NLP4RE - Identify NLP techniques used
Ambiguity in user stories: A Systematic Literature review (Amna & Poels (2022a))	Review the studies that investigate or develop solutions for problems related to ambiguity in user stories.	<ul style="list-style-type: none"> - How researchers define ambiguity problems? - Identify solutions for ambiguity in user stories - Analyze evidence of the effectiveness of the identified solutions

2.2 RESEARCH DESIGN

As mentioned, this study aims to better understand the current state of the art in automated support for generating user stories. To this end, it explored literature-available datasets, examined techniques for creating user stories, and comprehended how researchers evaluate user story quality. We followed [Kitchenham *et al.* \(2015\)](#) guidelines for SLR to identify and summarize available literature based on outlined RQs. In this Section, we present details about the RQs and report the SLR protocol steps performed in our study ([Kitchenham *et al.*, 2015](#)). In Table 2.2, we detail the rationale behind the consideration of each RQ.

Table 2.2 : Research questions and their rationales.

#	Research question	Rationale
RQ1	What datasets are used by researchers, and in what way, to generate user stories automatically?	This research question provides an understanding of the existence of specific dataset sources used by researchers to generate user stories automatically.
RQ2	Which techniques are used to specify user stories automatically?	This question looks for techniques and details on their application to specify user stories automatically.
RQ3	How user story quality is evaluated?	Provides an overview of how user stories have been evaluated by researchers regarding their quality.

In Section 2.2.1, we describe our search strategy and selection criteria adopted to retrieve and select published research on user story generation systematically. Section 2.2.2 relates the quality criteria analysis performed to evaluate the quality of the selected studies. Finally, in Section 2.2.3, we explain how we extracted and synthesized data to answer the proposed RQs.

2.2.1 SEARCH STRATEGY AND STUDY SELECTION

We adopted a hybrid (two-stage) search strategy combining database search with snowballing (Mourão *et al.*, 2020). First, we performed a database search (Kitchenham *et al.*, 2015) and then a snowballing search (Wohlin, 2014). A database search involves developing and running a search string in a digital library or digital library (Kitchenham *et al.*, 2015). A snowballing search analyses the references (backward) and citations (forward) of selected studies, aiming to detect other relevant studies (Wohlin, 2014). According to Mourão *et al.* (2020), the combination of database search from the *Scopus*³ digital library with forward and backward snowballing represents an appropriate alternative to be adopted as a search strategy for SLRs.

To perform the database search, we identified suitable keywords. The keywords definition was initially based on the authors' knowledge and past experiences (Zhang & Babar, 2013). Additionally, as suggested by Kitchenham *et al.* (2015), we reviewed our RQs and identified important concepts and terms used in the RQs related to the main topic of interest to generate our group of domain keywords. Next, we tested the capacity of the multiple combinations of the domain keywords to retrieve a set of studies known by the authors (control group) in *Scopus* digital library. As recommended by Kitchenham *et al.* (2015), the study control group was used to calibrate our search string. In the following, we present our full search string.

```
((“user stor*”) AND (processing OR automat* OR extract* OR “text  
predict*” OR generat* OR “natural language” OR nlp OR “machine learning”  
OR ml ))
```

³<https://www.scopus.com>

We ran our final search string on *Scopus* digital library in April 2024 in three metadata fields: title, abstract and keywords. The search string was also adapted to meet specific search criteria (e.g. syntax) of the digital library. Besides the recommendation of [Mourão et al. \(2020\)](#), we chose Scopus because it indexes studies of several international publishers, including *Springer*, *Wiley-Blackwell*, *Elsevier*, *IEEE Xplore* and *ACM Digital Library*. As a result of our database search, 637 studies were retrieved. Next, selection criteria were defined for deciding which documents retrieved were relevant. The selection criteria are organized into three Inclusion Criteria (IC) and Exclusion Criteria (EC):

- **IC1:** The study must present or mention the automated generation of use story adopting NLP and/or ML techniques; *AND*
- **IC2:** The study must be peer-reviewed.
- **EC1:** The study is not a journal or conference paper; *OR*
- **EC2:** The study is not written in English; *OR*
- **EC3:** The study is an older version of another study already considered.

We first performed cleaning in our dataset of 637 retrieved documents by removing conference announcements and documents that were out of the scope of the computer science domain. This stage resulted in 524 documents. Next, we applied the IC and EC based on the analysis of the title, abstract and keywords. Given the high number of documents to be analyzed, these criteria were manually applied by the corresponding author only. However, any doubt about the inclusion or exclusion of a study was discussed with other authors through consensus meetings. A total of 16 studies were selected as candidate studies at this stage. Lastly, the IC and EC criteria were applied based on the full-text analysis of the candidate

studies. 5 out of 16 studies were excluded, resulting in 11 included studies during the database search analysis.

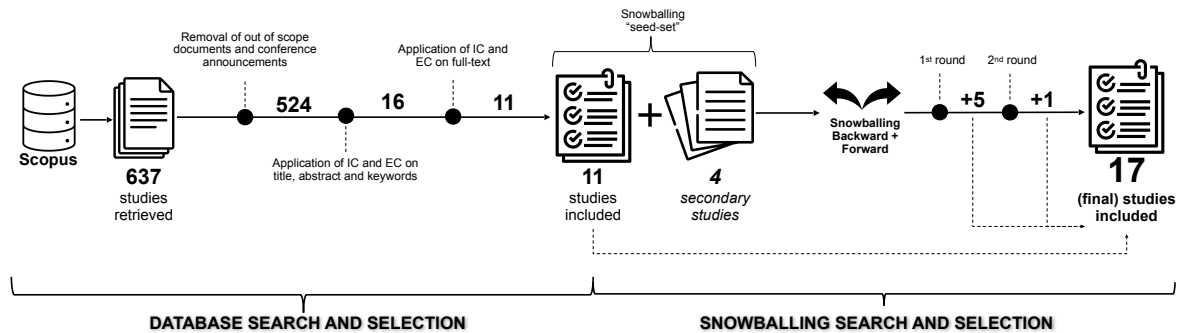


Figure 2.1 : Search strategy and selection process.

It is worth mentioning that during the analysis of the title, abstract and keywords of the studies retrieved from Scopus, four secondary studies (Raharjana *et al.*, 2021; Amna & Poels, 2022b; Cheliger *et al.*, 2022; Amna & Poels, 2022a) (SLRs and systematic mappings – a kind of lightweight SLR that aims to survey the available knowledge about a research topic (Kitchenham *et al.*, 2015)) were identified (listed in Subsection 2.1.1). Since they are not primary studies (e.g. case studies, empirical studies, surveys, etc.), we considered them separately as related work. They are described in Section 2.1.1.

Following the second stage of the hybrid search strategy (Mourão *et al.*, 2020), we perform forward and backward snowballing to avoid missing relevant studies. We followed the guidelines proposed by Wohlin (2014) to perform the snowballing search. We used the 11 included studies from our database search process + 4 secondary studies considered as related work as our “seed set” (starting set) for performing two iterations of the snowballing technique. We performed the forward and backward snowballing techniques in parallel (i.e. backward, forward, and so on). The studies’ citations were extracted with the support of

digital libraries, such as *Google Scholar*⁴ and *Scopus*. In each snowballing iteration, we first applied IC and EC criteria on the title, abstract, and keywords and then on the full text. We performed two backward and forward snowballing iterations, stopping their execution at the last iteration because no more relevant study was detected. One author performed the snowballing technique, and when doubts about the inclusion or exclusion of a paper were raised, it was discussed with other authors in meetings. As a result of both snowballing techniques, we added 6 (5 first rounds and 1 second round) more relevant studies to our set of included studies, totaling a final set of 17 included studies. The final list of the included studies is presented in Table 2.6. Figure 2.1 illustrates the search and selection process and its findings.

2.2.2 QUALITY ASSESSMENT

We followed the guidelines proposed by *Wieringa et al. (2006)* to assess the quality of the selected primary studies. This study recommends an evaluation criteria for different published studies in the RE field. We adopted the five Quality Assessment (QA) questions recommended for studies that present solutions for RE problems (Table 2.3). For each question, we assigned a score of (1) if the primary study explicitly addressed the QA question, (0.5) points if the question is partially addressed, and (0) for the QA questions that do not present related evidence in the study.

To facilitate the comprehension of our analysis, we presented the authors' rationale during the papers' analysis to answer each QA question. In Table 2.4, we present the final sum of the scores for each primary study selected. The scores given in Table 2.4 resulted from consensus between the authors.

⁴<https://scholar.google.com>

Table 2.3 : Quality assessment questions [Wieringa et al. \(2006\)](#).

#	Questions	Rationale
<i>Proposed-of-solution research</i>		
QA1	Is the problem to be solved by the technique clearly explained?	The scope of the problem must be delimited in the paper. Assess the level of understanding of the problem by the authors.
QA2	Is the technique novel, or is the application of the techniques to this kind of problem novel?	Level of novelty regarding the technique or solution presented in the paper in comparison with related work discussed by the paper's authors or existing literature until the paper's publication time.
QA3	Is the technique sufficiently well described so that the author or others can validate it in later research?	Perception at the level of replicability and reproducibility of the presented technique/solution.
QA4	Is the broader relevance of this novel technique argued?	The significance and potential impact of the technique/solution in a general view of the domain under investigation.
QA5	Is there sufficient discussion of related work?	The mention and the discussion of strengths and/or limitations of previous approaches and how the current paper is motivated by them or addresses these limitations.

The objective of QA is to ensure that the studies included in the SLR present an acceptable level of quality.

It is worth mentioning that the adopted criteria were simply guidelines rather than quantitative evaluations with standardized values. The evaluation of the papers was based strictly on the interpretation of the authors considering the rationale behind each question. [Amna & Poels \(2022a\)](#) also followed the QA procedures proposed by [Wieringa et al. \(2006\)](#). We present our QA results using the same format and criteria used for [Amna & Poels \(2022a\)](#).

Table 2.4 : Quality assessment scores.

Study	Year	QA1	QA2	QA3	QA4	QA5	Score	Qual.
Thamrongchote & Vatanawood (2016)	2016	0.5	1.0	0.5	0.5	0.0	2.5	50%
Rodeghero <i>et al.</i> (2017)	2017	0.5	1.0	1.0	1.0	1.0	4.5	90%
Murtazina & Avdeenko (2019)	2018	0.5	0.5	0.5	0.5	0.5	2.5	50%
Santos <i>et al.</i> (2018)	2018	0.5	0.5	1.0	0.0	0.5	2.5	50%
Raharjana <i>et al.</i> (2019)	2019	1.0	1.0	0.5	0.5	0.5	3.5	70%
Li <i>et al.</i> (2019)	2019	1.0	1.0	0.5	0.5	1.0	4.0	80%
Peña Veitía <i>et al.</i> (2020)	2020	1.0	1.0	0.5	0.5	1.0	4.0	80%
Resketi <i>et al.</i> (2020)	2020	1.0	1.0	1.0	1.0	1.0	5.0	100%
Henriksson & Zdravkovic (2020)	2020	1.0	1.0	1.0	1.0	1.0	5.0	100%
Panichella & Ruiz (2020)	2020	1.0	0.5	0.5	0.0	0.5	2.5	50%
Nistala <i>et al.</i> (2022)	2022	1.0	1.0	1.0	1.0	1.0	5.0	100%
Kumar <i>et al.</i> (2022)	2022	1.0	1.0	0.5	0.0	0.5	3.0	60%
Lam <i>et al.</i> (2022)	2022	1.0	1.0	0.5	0.0	1.0	3.5	70%
Dwitam & Rusli (2020)	2022	0.5	1.0	0.5	0.5	0.0	2.5	50%
Heng <i>et al.</i> (2023)	2023	0.5	0.5	0.5	1.0	1.0	3.5	70%
Mateus <i>et al.</i> (2023)	2023	1.0	1.0	1.0	0.5	0.5	4.0	80%
Siahaan <i>et al.</i> (2023)	2023	1.0	0.5	1.0	1.0	1.0	4.5	90%

As illustrated in Table 2.4, all 17 studies included by IC and EC were retained as included studies after the QA.

2.2.3 DATA EXTRACTION AND ANALYSIS

To extract all relevant data to answer our RQs, we created a data extraction form based on our RQ goals. We used this predefined data extraction form as specified in Table 2.5 to organize data extracted from the different primary studies. The data were extracted by the corresponding author and samples of the data were reviewed by a second author in order to validate the data extraction phase.

Table 2.5 : Data extraction form.

#	Study data	Description	Relevant RQ
1	Identifier	Unique ID for the study	Study overview
2	Title		Study overview
3	Authors		Study overview
4	Year		Study overview
5	Article source		Study overview
6	Type of article	Journal, conference, workshop, book chapter	Study overview
7	1st author country		Study overview
8	Application context	Industrial, academic	Study overview
9	Date of data extraction		Study overview
10	Research Goal		Study overview
11	Data	Datasets used to automatically generate user stories	RQ1
12	Data	Is the dataset used publicly available?	RQ1
13	Data	How the datasets are used to generate user stories?	RQ1
14	Techniques	Is machine learning used?	RQ2
15	Techniques	Is natural language processing used?	RQ2
16	Techniques	Which techniques are used to generate user stories?	RQ2
17	Validation	How user story quality is evaluated?	RQ3
18	Validation	Some specific user story framework is used?	RQ3
19	Challenge and limitation	What challenges and limitations did the study acknowledge?	Study overview
20	Future work	What future work did the authors suggest?	Study overview

The extracted data is publically available online⁵. This document presents the extracted data for each primary study selected in our SLR. Each tab of the data extraction document refers to the data extracted from one paper following the form structure proposed in Table 2.5.

The data synthesis was performed through a combination of content analysis from the extracted data, categorizing the findings into broad thematic categories (Cruzes & Dybå, 2011) as well as a narrative synthesis (Popay *et al.*, 2006). The extracted information was interpreted

⁵<https://doi.org/10.5281/zenodo.8395878>

and analyzed considering the authors' knowledge and experience in requirement elicitation and analysis. In the case of disagreements, consensus was established through discussion. The results of our analysis are described in Section 2.3 by answering the proposed RQs.

2.3 RESULTS

In this Section, we present the comprehensive findings of our SLR. The 17 selected papers are listed in Table 2.6, providing a concise overview of the specific studies selected in our study.

2.3.1 SUMMARY OF STUDIES

Of the 17 included primary studies in our SLR, 13 of them (76%) were published in conferences. The other four studies were published in journals (24%). We considered short papers in our datasets because they contribute essential to available approaches, though not all present rigorous validation steps. Of the studies selected, eight (47%) are full papers, and the rest (53%) are short papers. The documents were distributed in different venues, indicating no one venue's preference for publication on this topic. The graph illustrated in Figure 2.2 presents the number of journal and conference papers retrieved by year of publication.

The predominance of short papers and the lack of well-defined publication venues strongly suggest that this area is still in its early stages of development. In fact, it still needs more empirical evaluation of the approaches proposed as a way to validate if it remains an open and promising topic in RE. As user stories are short and simple, development teams are used to write them manually. The advancements in machine learning, language models and the conjunction of different NLP techniques can bring opportunities to automate the writing of user stories in software development projects.

2.3.2 (RQ1) WHAT DATASETS ARE USED BY RESEARCHERS, AND IN WHAT WAY, TO GENERATE USER STORIES AUTOMATICALLY?

The selection of valuable datasets is primary for the success of approaches that involve ML. It also makes it possible for other researchers to reach the same results by reproducing the experiment using the same dataset or comparing it with different approaches. In this Section, we identified the datasets used to specify user stories in the selected primary studies. We also investigated how they were used by the researchers to achieve that goal.

We identified 16 different software requirements datasets in our primary studies. However, only eight of them are publicly available and the access link can be found in Table 2.6. No attempt was made to contact them to gain access to the datasets, so we only share the access links provided by the authors along with their papers. The public datasets can be reused to validate different approaches involving the generation and validation of user stories in different contexts. The rather low percentage of public datasets among the studies

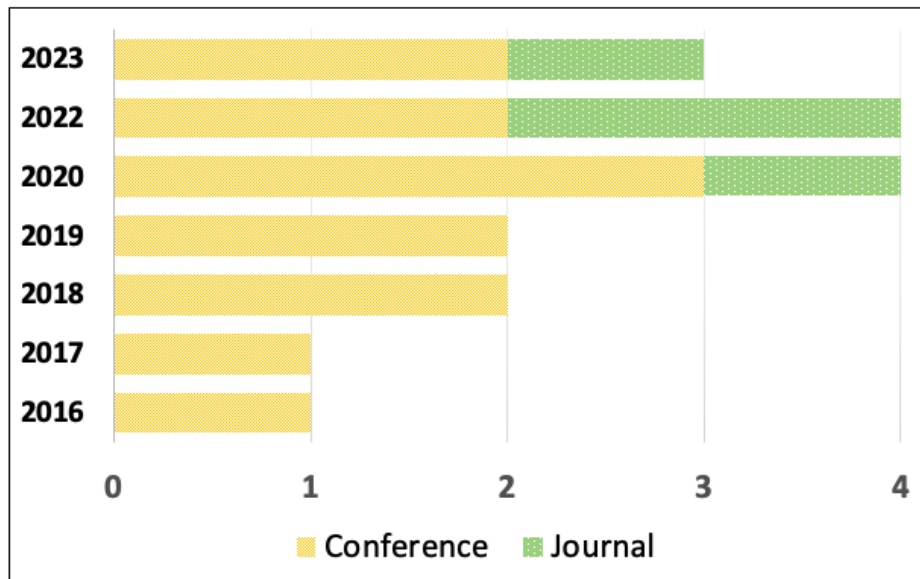


Figure 2.2 : The number of collected conference and journal papers until April 2024.

Table 2.6 : Primary studies and the datasets used to automatically identify and/or generate user stories.

Study	Year	Type	Size	Dataset	Available?
Thamrongchote & Vatanawood (2016)	2016	Conference	Short	Historical data	No
Rodeghero et al. (2017)	2017	Conference	Full	Transcripts	Yes
Murtazina & Avdeenko (2019)	2018	Conference	Short	-	No
Santos et al. (2018)	2018	Conference	Short	Use Cases UH4SP	No
Raharjana et al. (2019)	2019	Conference	Short	Online News Dataset	No
Li et al. (2019)	2019	Conference	Short	-	No
Peña Veitía et al. (2020)	2020	Conference	Short	Issue Tracker CONNECT	Yes
Peña Veitía et al. (2020)	2020	Conference	Short	Requirements datasets	Yes
Resketi et al. (2020)	2020	Journal	Full	Industry dataset	No
Resketi et al. (2020)	2020	Journal	Full	Trident project	Yes
Henriksson & Zdravkovic (2020)	2020	Conference	Full	-	No
Panichella & Ruiz (2020)	2020	Conference	Short	-	No
Nistala et al. (2022)	2022	Journal	Full	Industrial dataset	No
Kumar et al. (2022)	2022	Conference	Short	Requirements datasets	Yes
Lam et al. (2022)	2022	Conference	Short	Chatbot conversation	No
Dwitam & Rusli (2020)	2022	Journal	Full	Chatbot conversation	No
Heng et al. (2023)	2023	Conference	Full	US/BDD scenarios	Yes
Mateus et al. (2023)	2023	Conference	Full	-	No
Siahaan et al. (2023)	2023	Journal	Full	PURE	Yes
Siahaan et al. (2023)	2023	Journal	Full	Bloomsoft	Yes
Siahaan et al. (2023)	2023	Journal	Full	Wordnet	Yes
Siahaan et al. (2023)	2023	Journal	Full	Online News Dataset v2	No

may be partially explained by the nature of software engineering; companies tend to keep their development process tightly closed to the public ([Ferrari et al., 2017](#)). Note that we also included two lexical dictionaries in our list, Wordnet [Fellbaum \(1998\)](#) and Bloomsoft [Castillo-Barrera et al. \(2018\)](#), used by [Siahaan et al. \(2023\)](#) to create a dictionary of terms present in user stories statements.

[Rodeghero et al. \(2017\)](#) used transcripts of spoken conversations during requirements gathering meetings to identify user story information. This paper does not present an approach to generating user stories but only identifies user story data in conversation transcriptions. The dataset⁶ used in this research was manually annotated to extract these parts from the text. Then

⁶<https://groups.inf.ed.ac.uk/ami/corpus/>

they trained an ML classifier to automatically recognize function and rationale information in the corpus.

Also, [Peña Veitía *et al.* \(2020\)](#) contribute by sharing their two software requirements datasets⁷⁸ used to identify user stories in software's issues records. As in the previous paper, the authors do not generate user stories, but they use natural language processing and machine learning models to identify user story information in the software issue tracker corpus.

One of them⁹ was also used by [Kumar *et al.* \(2022\)](#) in their work about user story splitting. The authors presented an approach to split complex user stories into less complex ones. For example, user stories representing CRUD operations (Create, Read, Update and Delete) could be split into four user stories (create, read, update, and delete), allowing the development team to better address the coding and testing steps. The authors automatically identified the complex user stories using a Python script based on linguistic heuristics.

Then, [Resketi *et al.* \(2020\)](#) shared their dataset used to summarize user stories automatically. This dataset was derived from a project aimed at facilitating the generation of metadata for digital collections at Duke University libraries while concurrently establishing a digital repository for digitized materials.

[Heng *et al.* \(2023\)](#) shared the dataset containing user stories and Behavior-driven Development scenarios (BDD). They used an ontology to organize data and build user stories. They believe that the use of an ontology can produce higher-quality user stories.

In addition to the lexical dictionaries Wordnet [Fellbaum \(1998\)](#) and Bloomsoft [Castillo-Barrera *et al.* \(2018\)](#) used by [Siahaan *et al.* \(2023\)](#), they also used PURE, the Public Require-

⁷<https://data.mendeley.com/datasets/7zvk8zsd8y/1>

⁸<https://connectopensource.atlassian.net/issues/?jql=orderbycreatedDESC&startIndex=50>

⁹<https://data.mendeley.com/datasets/7zvk8zsd8y/1>

ments Documents Dataset to support the creation of the user stories terms dictionary. This is a well-known dataset already used by other authors to propose diverse techniques in RE.

On the other hand, eight other datasets used by the authors are not publicly available. It happens due to industrial rights protection or even because the authors might overlooked sharing their datasets. For example, [Thamrongchote & Vatanawood \(2016\)](#) based their approach validation on the historical data of a project, but did not publish the dataset. Notably, they used this data to populate an ontology, which is essentially a structured representation of knowledge within a specific domain ([Guarino et al., 2009](#)). In this case, the ontology contained terms and relationships related to user stories, aimed at facilitating their creation and reuse.

[Raharjana et al. \(2019\)](#); [Siahaan et al. \(2023\)](#) created online news datasets, however, they did not share them with the readers. The authors used semantic role labeling to extract the aspects of *who*, *what*, and *why* from online news datasets to automatically create user stories using different natural language processing techniques. According to the authors, the usage of specified online news corpus in RE can contribute to a better domain knowledge understanding. We found these studies particularly interesting because of the nature of the datasets used and how the proposed techniques were able to collect data from the news text.

[Mateus et al. \(2023\)](#) proposed user story generation by applying transformation patterns to BPMN graphs. They present a well-described example of transformation in their paper though they do not use any specific dataset to perform a more complete evaluation of the proposed technique.

It also draws attention to the other three short papers that do not use any dataset to validate their approach [Murtazina & Avdeenko \(2019\)](#); [Li et al. \(2019\)](#); [Panichella & Ruiz \(2020\)](#), as they are only concerned with explaining an idea or approach without deep data validation. Also, [Henriksson & Zdravkovic \(2020\)](#) did not use any datasets when proposing

Table 2.7 : Main techniques used for each primary study to identify and generate user stories automatically.

Study	NLP	ML	Main Technique
Thamrongchote & Vatanawood (2016)	Yes	No	Ontology
Rodeghero <i>et al.</i> (2017)	No	Yes	ML classification
Murtazina & Avdeenko (2019)	Yes	No	Ontology
Santos <i>et al.</i> (2018)	Yes	No	Language heuristics
Raharjana <i>et al.</i> (2019)	Yes	No	NER
Li <i>et al.</i> (2019)	Yes	No	Graph theory
Peña Veitía <i>et al.</i> (2020)	Yes	Yes	Neural Networks
Resketi <i>et al.</i> (2020)	Yes	No	Bag of words
Henriksson & Zdravkovic (2020)	Yes	Yes	NER
Panichella & Ruiz (2020)	No	Yes	ML classification
Nistala <i>et al.</i> (2022)	Yes	No	Text Parser
Kumar <i>et al.</i> (2022)	Yes	Yes	ML clustering
Lam <i>et al.</i> (2022)	Yes	No	Chatbot
Dwitam & Rusli (2020)	Yes	No	Chatbot
Heng <i>et al.</i> (2023)	Yes	No	Ontology
Mateus <i>et al.</i> (2023)	Yes	No	Language heuristics
Siahaan <i>et al.</i> (2023)	Yes	No	POS tagging

their approach for automated requirements elicitation, though the paper presents a transparent metadata model to collect software requirements data.

2.3.3 (RQ2) WHICH TECHNIQUES ARE USED TO SPECIFY USER STORIES AUTOMATICALLY?

For this study, we are considering using NLP and ML techniques for automatic user story identification and specification. Some papers also rely on manual tasks, such as natural language annotation and transformation rules. The goal of this research question is to identify the most used techniques and how they collaborate with the approaches proposed. Usually, a paper is going to combine different techniques to reach a goal. Table 2.7 lists an overview of the most important techniques or tools presented in the papers.

Ontologies were used by [Thamrongchote & Vatanawood \(2016\)](#); [Murtazina & Avdeenko \(2019\)](#); [Heng *et al.* \(2023\)](#) with different goals. The first paper uses ontologies to organize user story data for reuse in different projects. By doing that, creating new artifacts simpler and faster is possible. So, the second paper uses ontologies as a way to help the user evaluate the written quality based on ontology rules. Finally, the third one uses the dataset presented to populate an ontology and validate it. As a second step of this research, the authors intend to create a prototype tool to assist the writing of user stories.

Data collection plays an important role in the generation of user stories. We believe that the reasoning behind data collection and organization is part of the scope of this study. In this case, the studies [Thamrongchote & Vatanawood \(2016\)](#); [Murtazina & Avdeenko \(2019\)](#); [Heng *et al.* \(2023\)](#) take advantage of the user story template to organize data into ontologies serving as a database for diverse employments. It reinforces the simplicity of user story structure and how they are straightforward to employ.

[Lam *et al.* \(2022\)](#); [Dwitam & Rusli \(2020\)](#) used chatbot applications to support the elicitation step on requirements engineering. Firstly, [Lam *et al.* \(2022\)](#) used the IBM Watson chatbot system in a web application to gather clients' requirements and suggest user stories. They trained the chatbot in that way and presented all the aspects involved in this training. However, this solution is presented in the format of a short paper without a deeper analysis of its effectiveness. Also, the technology dependence might make it harder to replicate the study in other contexts.

Secondly, [Dwitam & Rusli \(2020\)](#) also used a chatbot for the same purpose. They relied on AIML (*Artificial Intelligence Markup Language*) to train the chatbot and made more use of natural language processing as a pre-processing step to deal with the Indonesian language. Both projects [Lam *et al.* \(2022\)](#); [Dwitam & Rusli \(2020\)](#) bring important contributions in terms

of using Artificial Intelligence (AI) in RE, though the solutions proposed are corpus-dependent and need different chatbot training to adapt them to different contexts.

[Raharjana et al. \(2019\)](#); [Henriksson & Zdravkovic \(2020\)](#) made use of NER (*Named Entity Recognition*) technique to deal with user stories. [Raharjana et al. \(2019\)](#) also explored the usage of Part-of-speech (POS) tagging combined with language heuristic. The processing steps were divided into role clustering, weighing and filtering to extract user story data from domain online news corpora. They proposed the composition of user stories by using language transformation rules.

Differently, [Henriksson & Zdravkovic \(2020\)](#) proposes a data model able to organize the RE process coming from different data sources. The data model presented can also store user stories converted from input data. To do that, they recommend a data extraction step based on classification algorithms, and sentiment analysis, in addition to NER. This is a theoretical paper not committed to presenting technical background details and therefore it is difficult to assess the feasibility of the approach in a realistic software development context.

[Siahaan et al. \(2023\)](#) prepared a study similar to [Raharjana et al. \(2019\)](#). They also used NLP techniques to identify user story data in online news datasets and compose user story statements using language heuristics. The techniques employed by the authors include part-of-speech (POS) chunking, named entity recognition (NER) and dependency parsing. However, they relied on lexical dictionaries (WordNet and BloomSoft) to support term identification related to software engineering.

[Rodeghero et al. \(2017\)](#) relied on machine learning classification to recognize user story information in team conversations. They first prepared a qualitative study to test the hypothesis that conversations between developers and customers contain role, function, and rationale information for user stories. Secondly, they performed a quantitative study to determine the

degree to which an existing classification algorithm can be trained to recognize this information in conversations. At the time of the study, they concluded that a Logistic Regression model was more prone to identify user story information in conversation corpora. They compared their results with the annotated corpus and they also used the standard machine learning performance metrics, such as Precision, Recall and others for evaluation. Nevertheless, since then, more advanced machine learning methods have emerged¹⁰ and with state-of-the-art methods, the results would probably improve drastically.

[Santos et al. \(2018\)](#) presented a theoretical approach using language heuristics capable of deriving logical architectures as a goal to define the initial requirements, in the form of user stories. They performed a controlled experiment deriving a module of architecture into a set of themes, epics, use cases and user stories. The authors believe they need to extend the number of transformation rules to cover other different types of requirements.

[Mateus et al. \(2023\)](#) also employed language heuristics to craft user stories. This study introduces an approach that streamlines the semi-automated generation of user stories and Gherkin scenarios from process models, utilizing transformation patterns. In this method, they utilize the Business Process Modeling Notation (BPMN) to represent business process models, and the transformation patterns are constructed based on the partial metamodels of BPMN.

[Li et al. \(2019\)](#) relied on graph definition using the Resource Description Framework (RDF) language and chatbot conversation to gather user stories. The paper proposes a conversation scheme able to transform the user's real life into a user story graph based on a series of first-order logic predicates. The usage of RDF language allows an easy transfer of learned knowledge into simulation environments by domain-specific controlling systems.

¹⁰<https://paperswithcode.com/task/text-classification>

Peña Veitía *et al.* (2020) worked to identify user stories in the issue tracker corpus. They trained two different binary models of Neural Networks using Tensor Flow¹¹ and executed a comparison between both. The first model is a Long Short-Term Memory (LSTM) bidirectional neural network and the second one is a pre-trained language model called ELMo (pet, 2018). The results identified the model ELMo as being the best fit for the problem posed highlighting the potential of Large Language Models (LLM) for this task (Gilardi *et al.*, 2023). It classified issues in user stories with an efficiency of approximately 96%.

Resketi *et al.* (2020) made use of BoW (Bag of Words) to extract keywords and verbs from a set of user stories and write new ones. They also used tokenization, filtering, stop-word removal, and verb parser. The paper has a complete evaluation procedure composed of three different experiments, the first one quantitative, and the other two qualitative. They reached 97% of micro F-measure and 93% of macro F-measure, which is promising and concluded by the expert's feedback that these new user stories can be used as the base user stories in future similar projects.

Panichella & Ruiz (2020) proposed the tool Requirements-Collector, a machine and deep learning based tool for automating the tasks of requirements specification. This tool made use of a wide range of algorithms for this purpose, such as J48 (C4.5), PART, NaiveBayes, IBk (KNN), OneR, SMO, Logistic, AdaBoostM1, LogitBoost, DecisionStump, LinearRegression and RegressionByDiscretization. They used stakeholder meeting transcripts to automatically generate user stories and classify user reviews. The preliminary results highlight its accuracy for requirements extraction, though this is an ongoing work that still needs more evaluation. In the end, this is an exciting example of using different ML techniques to identify and specify system requirements.

¹¹<https://www.tensorflow.org>

[Nistala et al. \(2022\)](#) proposed an approach for generating context-sensitive user stories from diverse specifications. This is a complete work committed to using a combination of NLP techniques to convert different formats of requirements specifications into user stories that can seamlessly be linked up to other tools. The process proposed is divided into five steps. The first one is pre-processing, able to remove typographical errors, style errors and taxonomic variation. So, they parse the documents using Open NLP Parser¹² and Docx4j¹³. Next, they make a post-processing step using a classification algorithm combined with regex and heuristics. Then, they populate a model able to generate user stories as output and create the knowledge query that will allow the user to consult the model.

[Kumar et al. \(2022\)](#) proposed an approach to split complex user stories, so that implementation can be done quickly reducing the development time of software. They used machine learning clustering to evaluate the cohesion between user stories and decide whether they need to be split into smaller ones. The algorithms chosen for this experiment are K-means and K-medoids, being that K-means give a more remarkable output than K-medoids in this case. They performed the user story splitting manually based on CRUD and data entry breaking approaches. It is possible to run the clustering procedure as many times as needed to evaluate the cohesion of a set of user stories. If the cohesion between user stories is good then implementation becomes also easy for developers.

Table 2.8 presents a list of NLP and ML techniques used in the primary studies identified by our SLR. There is a notable presence of linguistic heuristics and regex algorithms to deal with user stories. They take advantage of the user story templates to break down the sentences or retrieve data. Although there is a strong presence of NLP approaches, it is also interesting

¹²<https://opennlp.apache.org/>

¹³<https://www.docx4java.org/trac/docx4j>

to note that the authors used different ML algorithms, and chatbots were employed to support the user story elicitation process.

Table 2.8 : Natural language processing and machine learning techniques used in primary studies to identify and generate user stories automatically (NLP = Natural Language Processing, SL = Supervised Learning, UL = Unsupervised Learning).

Technique	Category	Primary studies
Bag of words	NLP	Resketi et al. (2020)
Chatbot	Tool	Li et al. (2019) ; Dwitam & Rusli (2020) ; Lam et al. (2022)
Classification	SL	Rodeghero et al. (2017) ; Henriksson & Zdravkovic (2020) ; Panichella & Ruiz (2020)
Clustering	UL	Kumar et al. (2022)
Docx4j	Tool	Nistala et al. (2022)
Graphs	NLP	Li et al. (2019) ; Mateus et al. (2023)
Language heuristics or regex	NLP	Nistala et al. (2022) ; Resketi et al. (2020) ; Thamrongchote & Vatanawood (2016) ; Santos et al. (2018) ; Raharjana et al. (2019) ; Lam et al. (2022) ; Dwitam & Rusli (2020) ; Mateus et al. (2023)
Lexical dictionaries	NLP	Siahaan et al. (2023)
NER	NLP	Raharjana et al. (2019) ; Henriksson & Zdravkovic (2020) ; Siahaan et al. (2023)
Neural Networks	UL	Peña Veitía et al. (2020)
Ontology	NLP	Thamrongchote & Vatanawood (2016) ; Murtazina & Avdeenko (2019) ; Heng et al. (2023)
POS tagging	NLP	Raharjana et al. (2019) ; Siahaan et al. (2023)
Text Parser	NLP	Nistala et al. (2022) ; Resketi et al. (2020) ; Siahaan et al. (2023)
Tokenization	NLP	Resketi et al. (2020) ; Raharjana et al. (2019) ; Dwitam & Rusli (2020)
Sentiment analysis	SL	Henriksson & Zdravkovic (2020)

Table 2.9 : Summary on how the quality of user story generated is evaluated by the primary studies. The symbol “-” means *none or unspecified*.

Study	Quality evaluation	RE framework or tool
Thamrongchote & Vatanawood (2016)	-	-
Rodeghero <i>et al.</i> (2017)	Quantitative	-
Murtazina & Avdeenko (2019)	Qualitative	Ontology
Santos <i>et al.</i> (2018)	-	-
Raharjana <i>et al.</i> (2019)	Empirical	-
Li <i>et al.</i> (2019)	-	-
Peña Veitía <i>et al.</i> (2020)	Quantitative	-
Resketi <i>et al.</i> (2020)	Qualitative	QUS/Survey
Henriksson & Zdravkovic (2020)	-	-
Panichella & Ruiz (2020)	-	-
Nistala <i>et al.</i> (2022)	Qualitative	-
Kumar <i>et al.</i> (2022)	Quantitative	-
Lam <i>et al.</i> (2022)	Qualitative	ISO 9126
Dwitam & Rusli (2020)	Quantitative	-
Heng <i>et al.</i> (2023)	-	-
Mateus <i>et al.</i> (2023)	-	-
Siahaan <i>et al.</i> (2023)	Quantitative	-

2.3.4 (RQ3) HOW USER STORY QUALITY IS EVALUATED?

Different ways exist to evaluate the user story quality as described in Section 1.2.1. The goal of this research question is to understand which techniques were applied by the authors to measure the overall quality of user stories automatically generated. Although some approaches are well-defined and used in academia, we wanted to verify whether the authors relied on them as an evaluation step in their works. Overall, the adoption of RE quality guidelines to evaluate user stories generated is low, though some papers made use of quantitative methods to evaluate the proposed approaches. Table 2.9 summarizes our findings.

Numerically, seven of the 17 studies (41%) (Thamrongchote & Vatanawood, 2016; Santos *et al.*, 2018; Li *et al.*, 2019; Henriksson & Zdravkovic, 2020; Panichella & Ruiz, 2020;

Mateus *et al.*, 2023; Heng *et al.*, 2023) did not present any type of quality evaluation. The major part of these studies are short papers only committed to presenting their approaches without validation. Five studies (29%) (Rodeghero *et al.*, 2017; Peña Veitía *et al.*, 2020; Kumar *et al.*, 2022; Dwitam & Rusli, 2020; Siahaan *et al.*, 2023) are based on only quantitative measures for validation, without running any controlled experiment with humans or RE guidance. In addition, the other four papers (24%) (Murtazina & Avdeenko, 2019; Resketi *et al.*, 2020; Nistala *et al.*, 2022; Lam *et al.*, 2022) used some qualitative approaches to check the characteristics of the user stories generated. Finally, one paper (6%) Raharjana *et al.* (2019) made only an empirical evaluation of the dataset of user stories resulting based on the author's experience in RE. It is also worth mentioning that Heng *et al.* (2023) performed a guided interview with software specialists to validate the ontology proposed for generating user stories and BDD scenarios.

Rodeghero *et al.* (2017) used the standard machine learning performance metrics (*precision, recall, true positive rate, false positive rate, and pyramid precision*) to evaluate the quality of the extracted data compared with the manually annotated data. By using these metrics the authors could find the best training configuration for user story data classification. Peña Veitía *et al.* (2020) followed a similar procedure to validate their machine-learning approach. In this case, the authors relied on neural networks to classify user stories.

Kumar *et al.* (2022) also used an automated statistical evaluation for their results. They compared the *silhouette value* between non-split and split user story sets to measure the cohesion among user stories. The higher the *silhouette value* higher the cohesion. Dwitam & Rusli (2020) used several metrics to evaluate the performance of the proposed chatbot, such as *total elapsed time, total number of user turn, and total number of system turn*, among others. They also surveyed to investigate how the users were prone to use the chatbot application

proposed by them. However, they do not evaluate the quality of the user stories generated by the chatbot application.

[Siahaan *et al.* \(2023\)](#) manually labelled semantic role based on the aspects of *who*, *what* and *why* in the news dataset. So, they performed the automatic extraction using the NLP techniques and based their evaluation on usual statistic measures of precision, recall and F-measure comparing the elements extracted with those manually annotated.

[Murtazina & Avdeenko \(2019\)](#) used specific rules based on a domain ontology to validate the user stories created. These rules validated the presence of the main elements of a user story: role, action and benefit. The ontological approach differs from others because it can be easily applied to new and existing artifacts to ensure the quality of the requirements. It is valuable for agile environments where changes are welcome in different steps of the development process.

[Resketi *et al.* \(2020\)](#) made use of the AQUASA tool (based on the QUS framework) as a preprocessing step to check the quality of the user stories used in their experiment and improve the user stories generated by their approach. The final set of user stories generated is evaluated statistically and by software practitioners through a survey. This paper brings a real concern about the user stories' quality considering their evaluation before and after the approach proposed.

[Nistala *et al.* \(2022\)](#) relied on the expert's experience to validate the feature and user stories produced by the data model proposed. The experts gave detailed feedback (scores and comments) based on a comparison of the generated output with the original data from the source specification document. The main focus of this analysis was to check the correctness of the extracted sentences.

Lam *et al.* (2022) utilized a controlled experiment to validate the user stories generated by their chatbot application. They used the survey approach based on ISO 9126 to measure completeness, correctness, and verifiability. The survey was designed using a 5-point Likert scale (0 = Totally disagree, 1 = Disagree, 2 = Neither agree nor disagree, 3 = Agree, 4 = Strongly agree), composed of five questions. They concluded that their system was able to create user stories more complete, correct and verifiable than using traditional RE techniques.

2.4 DISCUSSION

Software requirements can emanate from various sources and exist in different formats, reflecting the diverse nature of the development process (Aranda *et al.*, 2010). Stakeholders, such as clients, end-users, and domain experts, often play a crucial role in identifying functional and non-functional requirements. These requirements are gathered through interviews, surveys, workshops, and user feedback. While requirements manifest in various formats, user stories are one of the preferred methods of software developers because of their convenience and simplicity (Raharjana *et al.*, 2021).

Due to time constraints, inexperience, and other reasons discussed in this paper, manually defined user stories often lack quality and consistency (Wagner *et al.*, 2019; Kustiawan & Lim, 2023). High-quality user stories are mandatory to ensure effective communication and comprehension among the development team and stakeholders.

Through this SLR, we have confirmed the interest in research on automated writing of user stories. Albeit still in its infancy, this topic promises to improve software development in three significant ways. First, by employing efficient techniques and tools, teams can expedite the gathering of requirements from stakeholders, reducing the overall project duration. Saving time in RE and specification steps is essential for efficient and streamlined software

development processes. Second, assuming that the generating method is accurate, it would undoubtedly contribute to enhancing the overall quality and ensuring a more consistent writing style across the project. Last but not least, it would result in more time spent on the development instead of on refinement meetings. Consequently, teams could get faster feedback from their clients and be more agile in the broad sense.

2.4.1 RQ1 - DISCUSSING THE AVAILABILITY OF USER STORIES DATASETS

The small number of studies and research papers however give a glimpse of the headwinds faced to make this a reality. First, we have seen that very few datasets are publicly available to conduct such research. As we are shifting toward machine learning-based approaches, it is increasingly critical that researchers and practitioners share their datasets; both to further enhance repeatability and improve the quality of the research results.

Indeed, among the papers surveyed, most of them are short papers with limited validation or none at all. In some of them, the corpus is missing and in others, there is no experiment done to validate the approach. Therefore, it seems clear that generating user stories automatically is an open topic at the crossroads of RE and AI.

In ML, the requirement for bigger datasets is closely correlated to the data and the model complexity. Text is well-known as one of the highest dimensionality data occurring in real-world problems [Görnitz *et al.* \(2015\)](#). Small datasets may also hinder model generalization and provoke overfitting [Géron \(2019\)](#). Moreover, the lack of datasets slows progress, making it harder to evaluate ML models in different scenarios. In turn, superficial results prevent the adoption of those approaches by the RE community.

The resolution to the challenges we face may lie in adopting a transdisciplinary approach, one that intersects the realms of Artificial Intelligence (AI) and Requirements Engineering

(RE) research. Currently, there is a palpable excitement surrounding AI, which has sparked motivating interest within software development companies. This enthusiasm underscores the potential of AI tools.

Software companies have data warehouses of different types of requirements artifacts. For instance, project tracking software stores vast quantities of user stories, requirements, business rules, and other pertinent data. Leveraging these rich datasets could serve to empower the training of various machine learning models, thereby offering invaluable assistance to RE practitioners.

RQ1 Findings

- The creation of new publicly available datasets to support projects focused on generating user stories appears to be a research opportunity in the area.
- The authors utilized datasets of varied formats and sources, which proved instrumental in generating user stories. This versatility can be attributed to the inherent nature of user stories, which often stem from diverse sources across different stages of software projects.
- Notably, some papers publicly share their datasets [Rodeghero et al. \(2017\)](#); [Peña Veitía et al. \(2020\)](#); [Resketi et al. \(2020\)](#); [Kumar et al. \(2022\)](#); [Heng et al. \(2023\)](#); [Siahaan et al. \(2023\)](#).
- The identified datasets publically available are [Meetings Transcripts](#), [Software Issue Tracker CONNECT](#), [Requirements dataset \(user stories\)](#), [PURE](#) and [US/BDD scenarios](#).

2.4.2 RQ2 - DISCUSSING THE APPROACHES EXPLORED BY THE PRIMARY STUDIES

Regarding the approaches explored by the primary studies, it is possible to identify a variety of different techniques employed, many of them combined to achieve better results in automatically defining user stories. We classified the techniques used in four different groups: Natural Language Processing, Supervised Machine Learning, Unsupervised Machine Learning and Tools (Table 2.8).

PREPROCESSING METHODS

In terms of NLP, the papers surveyed explored a diverse set of techniques to identify and generate textual user stories. Absolutely all the papers that proposed some approach used natural language processing in at least one step of the process to automatically identify or generate user stories. NLP provides varied techniques that can be combined with language heuristics to reach different goals. It can also be used as a preprocessing step to machine learning approaches.

It is quite common to observe well-known NLP techniques such as Tokenization, POS tagging, Text parser, Named Entity Recognition (NER) and Bag of words being cross-applied for preprocessing text in different scenarios. Some papers relied on these techniques to break down the natural language and then propose different language heuristics to identify and produce user stories.

Tokenization is the act of breaking down the text into smaller units (e.g. words, letters, etc.). This technique is generally essential in NLP or ML and many libraries and frameworks offer this algorithm off-the-shelf, such as NLTK¹⁴ and spaCy¹⁵ (Hagiwara, 2020).

Part-of-speech or POS tagging is a process that involves tagging each token of a sentence with a part-of-speech tag. This tag represents the grammatical class of the token, as a verb, noun, preposition, postpositions, adverbs, and so on. These tags are based on an international standard called *universal part-of-speech tagset*¹⁶ (Hagiwara, 2020).

The text parser technique is used to analyze the structure of a sentence, and it is a compliment to the POS tagging. There are two types of text parsers, the *constituency parsing* and the *dependency parsing*. The first one is used to break down the structure of a sentence according to its grammatical classification. For example, we can divide a sentence into two parts: a noun phrase (NP) followed by a verb phrase (VP), and so break each part into smaller grammatical portions. The second type of parser can specify the grammatical relation among the tokens of a sentence, using directional arrows to indicate the structural dependency (Hagiwara, 2020). The combination of the text parser with the POS tagging can be useful to identify parts of the user story by their syntax meaning. A verb, for example, can represent the action part of a user story. Nouns can represent the role or even some chunk of the benefit.

Named Entity Recognition (NER) algorithms identify and classify named entities within text, such as people, organizations, locations, and titles. NER can be leveraged to enhance role identification in the user story composition.

¹⁴<https://www.nltk.org/>

¹⁵<https://spacy.io/>

¹⁶<http://realworldnlpbook.com/ch1.html#universal-pos>

The Bag-of-Words (BoW) technique is a method for representing text documents by analyzing the frequency of individual words, ignoring their order and, generally, their grammar. While it does not capture the meaning of sentences, BoW can be useful for identifying prominent themes and topics within a collection of text documents. Hence, they can be a starting point for writing valuable user stories using the highest-ranked tokens. It is also generally used in conjunction with a Term-Frequency-Inverse Document Frequency ([Jones, 1972](#)) to weigh the importance of words.

It is important to highlight that NLP techniques alone are not enough to extract and write user stories. Usually, it is necessary a combination of techniques to produce such results. The majority of the authors relied on their own algorithms supported by language heuristics and NLP techniques to identify, collect and write user stories. A cross-technique approach seems to be the best option for those using NLP to deal with user stories.

KNOWLEDGE REPRESENTATION

Using ontologies is a clever approach to organizing user story data. Informally, the ontology of a certain domain defines its terminology, concepts, classification, taxonomy, axioms and relations. In a nutshell, ontology is an important part of the knowledge about any domain ([Gaaevic et al., 2006](#)). However, they suffer from the well-established drawbacks of ontology-based AI; demand of domain experts, difficulties in setting universally accepted concepts and relationships, limited lexical coverage, and the difficulties in their upgrading ([Lastra-Díaz et al., 2019](#)).

As software can be seen as an abstraction of the real world, ontologies can play an important role in establishing the elements and properties of the domain to empower software engineers to better understand the context in which the software will be used. As user stories

are split into three different parts (role, action, benefit) it makes it easy to organize data into categories and reuse data to create new and different user stories. Ontologies have long been employed to support software requirements elicitation, as can be read in the study of [Kaiya & Saeki \(2006\)](#).

At the same level of knowledge representation, we found the use of semantic role labelling technique to identify the aspects of *who*, *what* and *why* in user stories. Semantic Role Labeling (SRL) is a natural language processing task that involves identifying the semantic roles of words or phrases in a sentence and assigning them to specific syntactic constituents. These roles represent the different functions that words play to the sentence's main verb, such as agent, patient, or instrument ([Tan et al., 2018](#)). [Raharjana et al. \(2019\)](#); [Siahaan et al. \(2023\)](#) relied on their specific algorithms to identify semantic aspects of user stories.

The user story graph proposed by [Li et al. \(2019\)](#) is an example of how user stories are connected and dependent. One of the tasks performed by the authors used first-order logic predicates to identify the relationship among user stories. A better understanding of these relationships can assist software engineers in splitting and prioritizing user stories while working in agile environments.

CLASSIC MACHINE LEARNING

Our machine learning analysis is divided into two segments: supervised and unsupervised learning. Supervised learning is the classic task of learning from examples with *ground truth* to recognize the errors of the model and revise it. Some studies [Rodeghero et al. \(2017\)](#); [Henriksson & Zdravkovic \(2020\)](#); [Panichella & Ruiz \(2020\)](#) performed text classification using a supervised approach to identify user story information in different requirements datasets. Identifying user story information can be seen as an elicitation procedure able to save the

efforts of the development team in this task. The referenced papers made use of different algorithms for this goal and also made different use of the classified data. The drawback of this approach can be the corpus dependency to train a reliable model able to correctly identify user stories in different projects. It might be necessary to include other algorithms in this equation to increase the accuracy of the model. For example, [Henriksson & Zdravkovic \(2020\)](#) also relied on NER to support the identification of roles, and sentiment analysis to identify the benefit of the user story.

The usage of sentiment analysis was cleverly proposed by [Henriksson & Zdravkovic \(2020\)](#) to identify the type of emotion or attitude expressed in a chunk of a user story, including the possible value: positive, negative, or neutral. Based on a corpus of system user reviews it is possible to identify the type of request made by the user. For example, a negative sentiment can indicate a system bug or a disagreement with some feature. In short, this analysis can further support the specification of new user stories.

Alternatively, unsupervised machine learning involves algorithms that learn patterns and structures without receiving explicit *ground truth*. They analyze unlabeled data samples to uncover hidden relationships, clusters, or anomalies within the data itself. For instance, [Kumar et al. \(2022\)](#) used ML clustering (*K-means* and *K-medoids* algorithms) as a tool to validate their approach to user story splitting. It is one more case in which unsupervised learning strategies can be used for model or approach evaluation. In addition, after reading this paper we concluded that the cohesion comparison between two or more sets of user stories can also help software practitioners in validating their consistency and boundary, satisfying quality attributes of ISO/IEC/IEEE 29148:2011 ([ISO, 2018](#)).

NEURAL NETWORKS

Neural networks are a family of ML models inspired by the structure and function of their biological counterparts. They became famous in the last decade due to their inherent ability to learn good representations of complex data [Pham et al. \(2022\)](#). The study of [Peña Veitía et al. \(2020\)](#) employed them to identify user stories in a software issue tracker dataset. This is the only paper that made use of neural networks for this purpose. As a complement, the authors compared the results produced by a Long Short-Term Memory (LSTM) neural network with the ELMo language model [pet \(2018\)](#). ELMo is one of the first Large Language Models (LLMs) to emerge in the literature. ELMo is based on bidirectional LSTMs. Like other LLMs, it is trained using massive amounts of unlabelled text through a collection of pretraining objectives to enable the model to *teach itself* about the text [Webber \(2023\)](#). In a nutshell, the goal of an LLM is to correctly predict the probability distribution of the next word, given a sequence of words. LLMs were mostly popularized and formalized, however, by two keystone papers: BERT [Devlin et al. \(2019\)](#) and GPT [Radford et al. \(2018\)](#). Both are trained in different ways and can also generate text differently.

A considerable amount of models are continuously introduced ([Fu et al., 2023](#); [Chowdhery et al., 2022](#); [Melis et al., 2020](#); [So et al., 2021](#)) coming to address text generation proposing diverse parameters. Many of these models allow the user to fine-tune using specific corpora and so teach the model to generate answers more context-sensitive. For example, we can see studies trying to take advantage of the language models to address problems in software engineering ([Schröder, 2023](#); [Sridhara et al., 2023](#); [dos Santos & Petrillo, 2021](#)), and so proposing different ways to fine-tune the models to work in this specific context.

Most specifically aligned with the context of this SLR, we see [Jain et al. \(2023\)](#) evaluating different LLMs to automatically summarize the requirements present in software contracts

in a language comprehensible to business analysts. This study concludes that Pegasus LLM generates the most accurate summaries with the highest ROUGE score as compared to other models. They also relied on the experience of different analysts to evaluate the summaries generated. From the point of view of these authors, the language models are sufficiently powerful to disrupt the research in RE and propose state-of-the-art approaches to assist software practitioners in their daily routines.

TOOLS

We classified chatbots as a tool as different hands-on mechanisms can be used to train chatbot systems. A chatbot is a software program designed to mimic human conversation, typically with a user. While not all chatbots incorporate AI capabilities, modern ones generally leverage conversational AI methods like NLP and LLMs to comprehend user queries and provide automated responses ([cha, 2024](#)). Indeed, in the past couple of years, new chatbots being developed usually rely mostly on new large language models. Different chatbot builders can be used to create chatbots for an application. [Lam et al. \(2022\)](#) for example used the [IBM Watson Chatbot](#) system empowered by a web implementation to gather user story information. Additionally, there are other builders available in the market that make easier the implementation of chatbots, such as [Dialogflow](#), [Qnamaker](#) and [Woebot](#).

The usage of a chatbot is an interesting approach to gathering requirements and then converting them into user stories (also seen in ([Rajender Kumar Surana et al., 2019](#))). The authors relied on the user story structure to organize the conversation flow driving the user to give the correct elements that will help the system propose the user stories.

[Docx4j](#) is another tool used by the authors to deal with requirements descriptions. This tool is an open-source (Apache v2) Java library for creating, editing, and saving OpenXML

files. [Nistala *et al.* \(2022\)](#) used Docx4j to manipulate Microsoft Open XML files, and then extract natural language requirements to create machine-processable models to generate user stories. Docx4j is an example of how the authors can employ different libraries to gather requirements in different formats. One of the big advantages of automatically generating user stories is the capacity of the system to understand the software requirements emanating from diverse sources.

RQ2 Findings

- Most studies rely on the user story structure to gather information and/or generate sentences [Thamrongchote & Vatanawood \(2016\)](#); [Murtazina & Avdeenko \(2019\)](#); [Raharjana et al. \(2019\)](#); [Resketi et al. \(2020\)](#); [Nistala et al. \(2022\)](#); [Kumar et al. \(2022\)](#); [Dwitam & Rusli \(2020\)](#); [Siahaan et al. \(2023\)](#).
- Natural Language Preprocessing techniques, such as POS tagging and text parsing are common ground among the studies and are effective in supporting the user story data classification.
- Machine learning classification algorithms can be used to support the data-gathering process to composite user stories [Kumar et al. \(2022\)](#); [Panichella & Ruiz \(2020\)](#); [Peña Veitía et al. \(2020\)](#); [Rodeghero et al. \(2017\)](#).
- Knowledge representation, such as ontologies and graphs can be used to organize user stories data [Thamrongchote & Vatanawood \(2016\)](#); [Murtazina & Avdeenko \(2019\)](#); [Li et al. \(2019\)](#); [Mateus et al. \(2023\)](#).
- Chatbots are an effective option for gathering user story information [Lam et al. \(2022\)](#); [Dwitam & Rusli \(2020\)](#).
- Generation of user story statements is mostly a combination of language heuristics and natural language processing techniques.

2.4.3 RQ3 - DISCUSSING THE EMPLOYMENT OF USER STORY QUALITY GUIDELINES

With regard to the quality of the user stories generated we concluded that the studies are rarely relying on qualitative standards for user stories. We understand that there is a high

number of short papers, and it might be one of the reasons why they do not perform any qualitative evaluation steps. However, would be recommended to at least cite the framework or guideline intended to be used to assess the final quality of the statements written through the approach proposed.

It is important to remember that bad requirements descriptions do not guide the development of good software products. For this reason, the practice of evaluating the quality of user stories is very widespread in the market by software practitioners as shown in (Lucassen *et al.*, 2016b). From our perspective, the adoption of standard guidelines for evaluating the quality of user stories is an essential procedure in papers proposing the automatic specification of user stories. Only through this would be possible to assess whether the approach proposed can generate unambiguous requirements statements.

We found that some studies (Rodeghero *et al.*, 2017; Peña Veitía *et al.*, 2020; Kumar *et al.*, 2022; Dwitam & Rusli, 2020) adopted quantitative measurement as a way to assess the user stories generated. Although they are not ideal as a final evaluation step, they are an important part of this process. The authors can rely on different guidelines and technical approaches (Lucassen *et al.*, 2016a; Wake, 2003; ISO, 2018) to assess the user stories in a qualitative means. Possibly a controlled experiment would be necessary to reduce the bias of the evaluation.

We would like to highlight two papers that rely on established guidelines to evaluate the quality of software requirements. Lam *et al.* (2022) benefit from a survey approach based on ISO 9126 to measure completeness, correctness and verifiability. Plus, Resketi *et al.* (2020) made use of a tool created to evaluate user stories quality: AQUASA tool (based on the QUS framework). This is the only work that relies on a predefined framework for user stories.

An empirical evaluation made by authors cannot be enough to evaluate the statements generated, but it is a valuable complement to statistical assessment. Empirical evaluations rely on the software practitioner's experience in writing requirements statements. Many specialists benefit from their own experience to evaluate the quality of user stories ([Lucassen *et al.*, 2016b](#)). Though, would it be enough to evaluate an automated approach to specify user stories? Further controlled experiments may be necessary to ensure the effectiveness of the approach. Another possibility could be the usage of ontologies to evaluate the overall quality of user stories, as made by [Murtazina & Avdeenko \(2019\)](#).

Therefore, we conclude that there is broad space to employ better qualitative guidelines when generating user stories automatically. Statistical evaluation may not be the better way to assess the final quality of user stories due to the importance of these statements to the software development lifecycle, and the consequences of bad requirements to the final software product.

RQ3 Findings

- [Lam *et al.* \(2022\)](#) benefit from a survey approach based on ISO 9126 to measure completeness, correctness and verifiability.
- [Resketi *et al.* \(2020\)](#) made use of the AQUASA tool (based on the QUS framework) to evaluate user story quality.
- Ontologies can be used to validate the structure of user stories generated [Murtazina & Avdeenko \(2019\)](#).
- Authors could use more qualitative measurement to ensure that the user stories generated can be used for software practitioners.

2.5 FUTURE WORK

Other related papers concluded that there is a high potential for the applications of AI in RE (Liu *et al.*, 2022; Zhao *et al.*, 2021). For example, Zhao *et al.* (2021) states that the adoption of NLP on RE is an active and thriving research area as it has garnered a considerable number of publications and extensive attention from various researchers. While Cheligeer *et al.* (2022) consider that ML can support requirements activities both theoretically and practically. At the same time, the study mapping performed by Amna & Poels (2022b) on user story research identified the advantages of using user stories in requirement elicitation, which include improved team productivity, software quality and faster delivery.

The automated generation of user stories appears to be a promising area of research; however, the studies identified in this SLR indicate that it is still in its early stages of development, as pointed out by the broad presence of short papers in our dataset. Overall, we detected that most authors are committed to improving the approach presented in their studies as future works. It demonstrates the broad range of AI techniques available for user story generation. Only three studies do not present any future work plans (18%) (Thamrongchote & Vatanawood, 2016; Rodeghero *et al.*, 2017; Murtazina & Avdeenko, 2019).

The lack of reliable and openly accessible corpora for assessing methodologies across various scenarios exemplifies the vast potential yet to be explored in this field. Researchers can greatly enhance replicability by sharing their requirements corpora with the community. Of course, they can also work with the focus of creating new corpora and datasets in this area.

It is important to note that no standardized format for corpora is used in producing user stories (RQ1). Given that user stories can originate from a variety of sources, it is essential to employ approaches capable of collecting user story information from diverse channels and organizing it for subsequent text generation. Raharjana *et al.* (2019); Siahaan

et al. (2023) for example were able to retrieve user story context from online news datasets, while Peña Veitía *et al.* (2020); Kumar *et al.* (2022) used the same dataset of user stories in two different approaches. Raharjana *et al.* (2019) proposed an extension of their study by exercising the model on various online news sources and with different case events. Peña Veitía *et al.* (2020) also proposed refining the employed dataset by increasing the number of cases and retraining the model to have more accurate results. By doing that, they will prove the scalability, reliability and accuracy of the proposed models; we believe that researchers could reuse known corpora in RE, such as PURE¹⁷ and PROMISE¹⁸ or even work to create their corpora using data gathered from project management systems.

In terms of approaches employed (RQ2), we can see a variety of techniques being used to automatically generate user stories. It demonstrates how researchers use available AI techniques to deal with RE problems, from pure NLP approaches to deep learning strategies. LLM is being largely adopted in different scenarios (Ronanki *et al.*, 2023b; Scoggin & Torres Marques-Neto, 2024). We believe that new studies can take advantage of these models to complement their approaches or even develop new language models specific to user stories' context. We also found promising the usage of ontologies to organize user story data (Thamrongchote & Vatanawood, 2016; Murtazina & Avdeenko, 2019; Heng *et al.*, 2023) and the usage of lexical dictionaries to support the identification of user story data in datasets (Siahaan *et al.*, 2023).

Due to the general lack of industrial evaluation (Zhao *et al.*, 2021) researchers could work to implement their approaches in a way that software practitioners benefit greatly from daily. The utilization of portfolio management tools, such as Jira¹⁹, is common ground in

¹⁷<https://zenodo.org/records/7118517>

¹⁸<https://zenodo.org/records/268542>

¹⁹<https://www.atlassian.com/software/jira>

the software engineering market. Many of these tools offer the possibility of implementing plugins able to facilitate daily operations. In this sense, software engineers could take a lot of advantages of plugins able to automatically write user stories for them, and so proportionate industrial feedback to studies in the intersection of AI and RE.

Finally, the quality of the user stories generated is a cornerstone of this topic (RQ3). Some researchers did not evaluate the quality of their outputs as rigorously as could be desired. The low adoption of strict guidelines to evaluate user story quality demonstrates that researchers need to investigate well-known frameworks, tools and guidelines for this purpose. The authors could extend the papers published using more strict procedures for quality evaluation.

For quality evaluation, we recommend the usage of two frameworks designed to evaluate the quality of user stories: QUS framework ([Lucassen *et al.*, 2016a](#)) is based on 13 language quality criteria and INVEST ([Wake, 2003](#)) brings well-defined guidelines to assess the quality of user stories; we also point to the investigation of the Requirement Smells study proposed by [Femmer *et al.* \(2017\)](#), where are defined language criteria to the identification of bad software requirements; completely, authors could rely on the ISO standard ISO/IEC/IEEE 29148 [ISO \(2018\)](#) what defines quality attributes to requirements statements.

2.6 THREATS TO VALIDITY

In this Section, we enumerate the main threats to the validity of our study. We present our threats analysis based on well-known literature reference ([Wohlin *et al.*, 2012a](#); [Zhou *et al.*, 2016](#)).

Internal Validity. It might be possible that relevant primary studies are missed during the database search process. To mitigate this threat, we performed a hybrid two-step

search strategy as recommended by Mourão *et al.* (2020) involving both keyword-based and snowballing searches (Section 2.2.1). Furthermore, we tested and refined our search string as suggested by Kitchenham *et al.* (2015) to include only relevant terms. We opted to define a broader search string to obtain a more significant number of primary studies. The search and selection of studies as well as the data extraction process were performed by the corresponding author. As recommended by Wohlin *et al.* (2012a), interpretation problems or doubts were discussed with the other authors through consensus meetings.

External Validity. Regarding generalization of the results presented in our study, we followed well-established guidelines Kitchenham *et al.* (2015) to perform SLR and to obtain the most appropriate search strategy (Wohlin, 2014; Mourão *et al.*, 2020) aiming to maximize generalization. In Section 2.2, we detail our SLR protocol and make available online the selected studies and data extracted from them enabling the extension of this SLR and consequently further mitigation of this threat.

Construct Validity. A possible construct validity could be related to the data manipulation activities such as the data extraction process and insufficient or incorrect search strategy. Aiming to mitigate this threat, we include in our SLR protocol different measures such as the development of a data extraction form to mitigate inconsistencies and extraction bias, the definition of IC and EC and the performance of quality assessment of the selected studies.

Conclusion Validity. This threat addresses the reasonability and credibility of the results and conclusions presented in this study. To mitigate this threat we systematically created the data extraction form emerging from the RQs (Table 2.5). We performed a broad thematic analysis and narrative synthesis (Cruzes & Dybå, 2011; Popay *et al.*, 2006) to interpret and synthesize the extracted data to answer our RQs, and we conducted several meetings to discuss

study findings and draw the correct conclusions. Further research can be performed to evolve the results and conclusion reported in this study.

2.7 CONCLUSION

This SLR provides an overview of the current research on automatically generating user stories for software development. User stories are a prior approach to gathering software requirements in agile environments and can also be written relying on requirements artifacts. They are simple, easy to employ and able to embrace changes during the software development process, though they need to be well-written to communicate the right message to the development team.

Our SLR retrieved 17 papers addressing the automatic generation of user stories (as presented in Section 2.2). We evaluated all the documents as means to identify (i) which and how the studies used requirements corpora, (ii) which approaches were proposed and how they used ML and NLP to reach their goals, (iii) how these studies took care of the quality of the user stories generated.

We concluded that suitable corpora are scarce for experiments involving user stories. This makes it more complicated to develop new ideas and also tricky to replicability of the approaches proposed. Also, we identified a broad range of techniques to generate user stories automatically. However, the usage of well-known concepts of NLP and linguistic heuristics is presented in almost all studies. Finally, we close that the studies can use better guidelines to validate the quality of the user stories generated, though many of them propose quantitative metrics as evaluation. Despite what has already been accomplished, the best is yet to come.

As a future work, we could extend this review to include industry reports. Industry reports enhance the relevance and applicability of systematic literature reviews in software

engineering. While academic literature provides theoretical foundations and controlled experimental results, industry reports offer insight into real-world practices, challenges, and emerging trends that may not yet be captured in peer-reviewed publications. [Garousi *et al.* \(2019\)](#) encourage the use of diverse evidence sources to improve the external validity of findings.

CHAPTER III

AI-DRIVEN USER STORY GENERATION

3.1 INTRODUCTION

User stories are pivotal in agile software development, primarily due to their organized structure and the simplicity with which they can be executed. Nevertheless, development teams encounter the arduous task of dealing with the assortment of information expected from diverse sources to write user stories manually. This undertaking consumes a significant amount of time and is susceptible to inaccuracies. In this study, we automatically generated user stories using two distinct approaches: N-gram representation with linguistic heuristics and the GPT-3 model. We evaluated our work using diverse user stories corpora and calculated the metrics BLEU, ROUGE-N, and BERTScore for each set of user stories generated by both approaches. For the N-gram approach, we achieved an average: ROUGE-N=0.39, BLEU=0.26, BERTScore=0.73; and for the GPT-3 model: ROUGE-N=0.46, BLEU=0.27 and BERTScore=0.69. We concluded that although GPT models excel in producing more comprehensive user stories, N-gram models exhibit a higher level of semantic sensitivity. Considering its simplicity and minimal processing requirements, we recommend employing the N-gram technique for user story generation.

This study ([Dos Santos *et al.*, 2024](#)) was published in the *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)* in February 2024, entitled *AI-Driven User Story Generation* ([DOI 10.1109/ACDSA59508.2024.10467677](#)).

3.2 USER STORY GENERATION

We employed two different models to generate user stories. The first is an N-gram model supported by linguistic heuristics, and the second relies on a fine-tuned GPT model (OpenAI, 2024). The corpora used for this research were initially introduced by Dalpiaz *et al.* (2019) in their study about terminological ambiguity in user stories (the other six corpora were not used in this experiment as they were not publicly available).

The size of each corpus varies between 51 and 138 user stories. A total of 10 sentences of each corpus was separated for the evaluation step described in the Session 3.3. We separated them manually, trying to get different sorts of statements to run our tests, focusing on different roles and actions. The rest of the sentences were used to build context-based models for each corpus of user stories. Thus, we built 22 N-gram models and fine-tuned 22 versions of the GPT model for this study. Each model contains only tokens presented in its user stories.

The following subsections present the details of our model implementation.

3.2.1 N-GRAM MODEL

The proposed approach is based on a text generation system built using a statistical N-gram model (better described in Section 1.3). The choice for this method is based on the models' evaluation study produced by Hamarashid *et al.* (2022). This statistical model utilized a user stories corpora sample (Dalpiaz *et al.*, 2019) to aid software professionals in user story writing.

Figure 3.1 depicts this process. We designed this pipeline using NodeJS and established persistence with MongoDB to support our experiments. Firstly, about 90% of the sentences of each corpus are added into a database collection using a seed operation. Secondly, every user

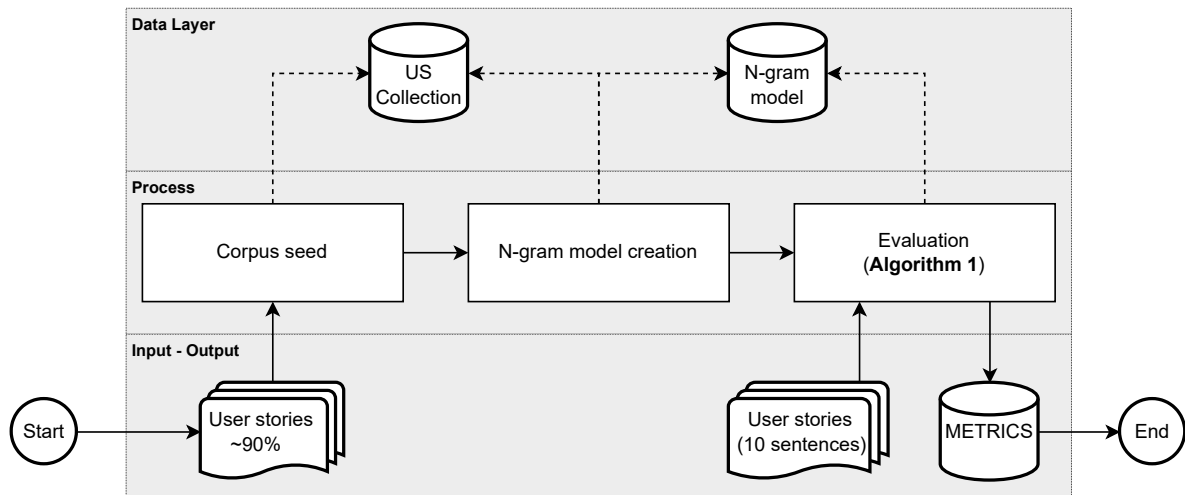


Figure 3.1 : Workflow diagram of the experimental protocol.

story added to the collection automatically updates the N-gram model with new tokens and their frequencies. We used the Natural JS library²⁰ to separate the user stories into N-grams. Next, they are sorted by frequency and stored in a separate MongoDB collection using the following structure: *input* (N-gram), *output* (1-gram) and *count* (number of times the input and output appear on the text).

The N-gram collection works as a database of text completions. Any part of the user story can be completed with the content of this collection. Given a random input, as "*I want to*", it is possible to find options to complete the sentence. The same works when the input is shorter or larger, as "*I*" or "*I want to have*". The N-gram model created is calculated using $n=9$. We chose this number arbitrarily, but in the end, we noticed it was a good choice for creating this statistical model. The N-gram model can receive up to eight tokens as input and complete it with one token.

²⁰<https://naturalnode.github.io/natural/>

3.2.2 GPT MODEL

The process to fine-tune a GPT model is well-described in the documentation²¹ and there are several other tutorials available teaching the best practices for training and data preparation. There are numerous advantages in fine-tuning a GPT model, such as getting higher quality results than prompting and lower latency requests.

Currently, there are two different formats of data input to fine-tune a GPT model. The chat format and the text-generation legacy mode. We opted for the last one as we are not training a conversational chatbot, but a text generator. We have to design data for this procedure based on a key pair of *prompt* and *completion*. These are some examples of training data used to fine-tune our model:

```
"prompt":"As", "completion":" a"
```

```
"prompt":"I want", "completion":" to"
```

```
"prompt":"I want to", "completion":" be"
```

We fine-tuned the GPT model using as input unigrams, bigrams, and trigrams of our N-gram data collection. To do that, we created a JSON file with our completions and input it into the OpenAI Platform using the model *davinci-002* as a base model. The training process takes around 30 minutes for each corpus to complete and it gives us a model ID as output that can be used to generate text via API or even directly on the OpenAI Playground.

²¹<https://platform.openai.com/docs/introduction>

3.3 EVALUATION

We simulated the generation of user stories using both proposed models. The idea behind this experiment is to mimic how the user could write a new user story being assisted by the models created. We believe that new user stories could be automatically added to both models to be reused in further elicitation sessions.

3.3.1 N-GRAM MODEL

To assess the N-gram model, we coded a NodeJS function (presented below) to automate our analysis. This function has as an input the model ID (from 1 to 22) that we would like to perform the evaluation on. So, we clean the database and run the seed operation to create the user stories and N-gram collections in the MongoDB instance.

As a second step, we retrieve the sample of testing user stories we would like to write using the model in memory. We split each user story into an array of tokens, and based on each sequence of tokens, we try to find the next one using the N-gram model. As our model was created using 9-grams, we can input into the model the top 8-grams to retrieve the most probable 9th-gram. This function works considering the best five options suggested by the model. It means that if the expected option is between the five first suggestions, it is selected and written in the sentence. If not, we still try to break the token into characters and use the first character of the token to retrieve the next token. If it is not found, we leave the function and present the sentence built as it is. We arbitrarily chose the cut-off of five suggestions, as we believe that the user will hardly scroll down far from five options to look for its desired token.

```

Data: Corpus Id
Result: USs Generated
1 remove all records from US collection;
2 remove all records from N-gram collection;
3 seed the model based on the Corpus Id;
4 testUSs ← loadTestingUserStories(CorpusId);
5 foreach testUS ∈ testUSs do
6   sentence ← [...];
7   tokens ← tokenize(testUS);
8   append token to the sentence;
9   foreach token ∈ tokens do
10    append a token to the sentence;
11    if current token is not the last in tokens then
12      search for N-gram records matching the last eight tokens, ordered by count
13      (desc), and limit to 5 (recursively until 1-gram);
14      if result NOT contains a record with an output equal to the next token then
15        search for 1-gram records matching the first character of the next token,
16        ordered by count (desc), and limit to 5;
17        if result NOT contains a record with an output equal to the next token
18        then
19          break out of the loop
20        end
21      end
22    end
23  append the sentence to output
24 end
25 return output

```

Algorithm 3.1: N-gram evaluation function

The output of the evaluation function is a set of user stories entirely or partially written using the N-gram model proposed. For example, we tried to mimic the writing of the following sentence: *As a user I want to search and discover music based on other users similar to myself.* Our model was able to write this sentence partially using always the first five completing options suggested by our model: *As a user I want to search and discover music based on other.* Even partially, the model proved capable of writing most of the sentence, though in other rare cases, it could complete only two tokens correctly.

We calculated the metrics BLEU, ROUGE-N, and BERTScore for all the sentences generated and compared them with the reference sentences. It gives us a quantitative result of our experiment. It helps us decide if we could proceed with a qualitative experiment with participants later to evaluate the feasibility of this approach. All the results reached on this experiment are available online on Zenodo²². Table 3.1 presents quantitative results for each corpus processed, with the best and worst scores highlighted.

We calculated ROUGE-N considering $n=2$, or bigrams (two-word sequences) to capture more context and specific phrases. We can assess the presence of particular words and expressions in the generated user story. If using $n=1$, we could only capture the most basic and general information, and overall text coherence and basic content overlap. The minimal score for ROUGE-N was 0.24, the maximum was 0.8, and the average was 0.39. For BLEU, the minimum was lower at 0.1, and the maximum stood at 0.75. BLEU's average score was lower at only 0.26. Finally, we can see that the BERTScore was significantly higher, with the lowest score being 0.65, the highest at 0.91, and the average at 0.73.

²²<https://doi.org/10.5281/zenodo.10044766>

Table 3.1 : Quantitative Metrics for N-Gram Model.

Corpus	Rouge	Bleu	BERTScore
c1	0.43	0.32	0.73
c2	0.26	0.14	0.65
c3	0.29	0.12	0.69
c4	0.37	0.23	0.73
c5	0.39	0.26	0.73
c6	0.32	0.15	0.7
c7	0.34	0.2	0.72
c8	0.45	0.33	0.77
c9	0.27	0.1	0.69
c10	0.31	0.16	0.72
c11	0.36	0.25	0.68
c12	0.46	0.32	0.74
c13	0.31	0.18	0.7
c14	0.46	0.38	0.74
c15	0.24	0.11	0.65
c16	0.8	0.75	0.91
c17	0.53	0.45	0.79
c18	0.33	0.17	0.71
c19	0.49	0.38	0.78
c20	0.31	0.16	0.71
c21	0.32	0.17	0.72
c22	0.48	0.38	0.79
AVG	0.39	0.26	0.73
MIN	0.24	0.1	0.65
MAX	0.8	0.75	0.91

3.3.2 GPT MODEL

For the GPT model, our evaluation was a bit more tricky. Trying to reproduce the same protocol as the N-gram model makes very little sense. Indeed, LLMs are better suited at generating complete answers or sentences than being used word by word like an autocomplete tool. Nevertheless, we tried it on several sentences to see how it would perform. In that case, we would send the input to GPT and check if the next word was among the top 5 proposed by the model. If it were not, the user story generation would be terminated. Perhaps

unsurprisingly, this process did not yield good results. We used only the top 5 since our goal is to simulate a user's behaviour who we believe would be unlikely to check more than five options.

Considering that we could not do the same experiments as with N-gram, we opted for a methodology more adapted to LLM. We used the OpenAI Playground to generate user stories using our fine-tuned models. To compare, we asked the model of each corpus to create a user story statement based on the following input pattern: "*As a role*". The role was selected based on the testing user stories dataset, so by doing that, we could better guide the GPT model in writing a sentence similar to those expected on the testing dataset.

Then, we stored the generated sentence beside the reference user stories in spreadsheets for each corpus. For example, for the reference user story "*As a user I want to be able to enter my zip code and get a list of nearby recycling facilities so that I can determine which ones I should consider*", we generated the sentence "*As a user I want to know the status of the recycling center so that I can know which centers are available around my area*".

Similarly to the N-gram model, we calculated the ROUGE-N, BLEU, and BertScore metrics for each generated sentence and made them available on Zenodo²³. The Table 3.2 shows the results we obtained.

Note that the minimal score for ROUGE-N was 0.34, the maximum was 0.57, and the average was 0.46. For BLEU, the minimum was lower at 0.18 and the maximum reached 0.34. BLEU's average score was lower at only 0.27. Finally, we can see that BERTScore was significantly higher with the lowest score at 0.61, the highest at 0.73, and the average at 0.69.

²³<https://doi.org/10.5281/zenodo.10044766>

Table 3.2 : Quantitative Metrics for GPT Model.

Corpus	Rouge	Bleu	BERTScore
c1	0.42	0.29	0.71
c2	0.52	0.34	0.70
c3	0.47	0.27	0.70
c4	0.48	0.29	0.71
c5	0.46	0.25	0.68
c6	0.47	0.25	0.67
c7	0.49	0.29	0.71
c8	0.57	0.31	0.70
c9	0.43	0.24	0.65
c10	0.51	0.27	0.69
c11	0.43	0.24	0.61
c12	0.48	0.24	0.68
c13	0.44	0.22	0.68
c14	0.50	0.27	0.70
c15	0.52	0.30	0.70
c16	0.47	0.31	0.71
c17	0.50	0.26	0.71
c18	0.34	0.18	0.66
c19	0.57	0.30	0.73
c20	0.38	0.20	0.68
c21	0.40	0.24	0.66
c22	0.35	0.28	0.71
AVG	0.46	0.27	0.69
MIN	0.34	0.18	0.61
MAX	0.57	0.34	0.73

3.4 DISCUSSION

We selected ROUGE and BLEU metrics (explained in Section 1.3.3) to compare the quality of generated user stories with the reference dataset at a syntactic level, focusing on evaluating the correctness of token sequences. Additionally, we incorporated the BERTScore metric to perform a semantic analysis of our generated user stories. This combination of three metrics enabled us to determine which technique excels in producing complete and semantically accurate user stories compared to our reference dataset.

The results indicated that the GPT models slightly outperformed the N-gram model in terms of ROUGE and BLEU statistical metrics. GPT models were able to generate more comprehensive user stories, contributing to their higher scores in these areas. However, when we considered the BERTScore, the N-gram model demonstrated better performance. We concluded that while GPT models excel in producing complete sentences, they do not reach the same level of semantic accuracy as the user stories generated by the N-gram model.

This may appear surprising. However, it is important to consider the nature of the task. User stories are typically structured and repetitive, which may not fully leverage the sophisticated generalization capabilities of a Large Language Model (LLM). LLMs excel at generating human-like language and semantic generalization, but for this particular task, the simplicity of the N-gram model proved effective. It can be quickly trained on a small yet specialized corpus, making it an ideal choice for user story generation across different branches of software development.

In the future, conducting experiments with real software developers is essential to validate these results and gauge potential productivity gains. In this context, the N-gram model could serve as a background tool to assist software development teams in completing text when writing user stories.

Another avenue for validation involves training the N-gram model on different corpus patterns. Since software requirements can originate from diverse sources in various formats, development teams must read, analyze, and create user stories based on client requirements. By extracting and storing N-grams from these corpora, we can aim to expedite the analysis process during meetings and ensure creating a more cohesive set of user stories.

We should also explore the optimal size of the N-gram model. Expanding the model to store more user stories might result in increased processing time or more false positive

completions. We can enhance our N-gram selection based on the user story template to address this. We can break down the user story into parts (role, action, and benefit) and store more precise N-grams in the model by employing text parsers and heuristics. Furthermore, developing more specific user story corpora for individual project initiatives can help the model maintain accuracy based on the project's unique textual context.

3.5 CONCLUSION

We conducted experiments employing N-gram and fine-tuned GPT models to create context-based user stories. Our assessment of these generated user stories incorporated ROUGE, BLEU, and BERTScore evaluation metrics, examining their syntactical and semantic aspects. Our findings indicated that while GPT models excelled in generating more comprehensive user stories, N-gram models demonstrated a higher degree of semantic sensitivity (as reflected in the BERTScore). Given the minor discrepancy in syntactic analysis and the difference in computational demands between the techniques, we chose the N-gram model as the preferred standard to support development teams in user story composition.

We also explored potential avenues for future research. This includes the consideration of a controlled experiment aimed at assessing the N-gram model's utility in aiding software practitioners in real-world user story creation scenarios. Additionally, we discussed enhancements to the model, such as its adaptability to training on diverse data formats and varying sizes, all while maintaining its accuracy.

CHAPTER IV

LEVERAGING TEXT GENERATION FOR ENHANCED USER STORY QUALITY: A CONTROLLED EXPERIMENT

4.1 INTRODUCTION

The approach employed to generate user story text in this chapter is an upgraded version of the one used in our previous study ([Dos Santos et al., 2024](#)) (Chapter 3). Based on our quantitative results, we recommended employing the N-gram technique for user story generation. In this study, we focused on creating N-grams that are more aligned with the user story template. We aim to validate this new N-gram model using a more strict quality evaluation framework for user stories. This section presents our N-gram model approach, followed by the prototype tool built to assist participants in writing user stories in a controlled experiment.

This study was submitted to the *Requirements Engineering Journal* and has been going through pair review since January 2025.

4.2 ADVANCING USER STORY TEXT GENERATION WITH N-GRAM MODELS

We propose a text generation system built using a statistical N-gram model. We relied on the theory behind employing N-gram models defined by [Jurafsky & Martin \(2009\)](#). Another author, [Hamarashid et al. \(2022\)](#), evaluated text prediction models and concluded that the N-gram model is a straightforward and more uncomplicated option for text prediction systems. Still, it gives an acceptable level of accuracy when using trigrams or four grams. We concluded that in our previous published paper ([Dos Santos et al., 2024](#)). In this chapter, we extracted parts of a corpus of user stories to create N-grams aligned with the user story template. The

corpus used for this research was initially presented by Dalpiaz *et al.* (2019) in their study about ambiguity in user stories (part of this corpus was not used in this experiment as it was not publicly available). Figure 4.1 depicts the process.

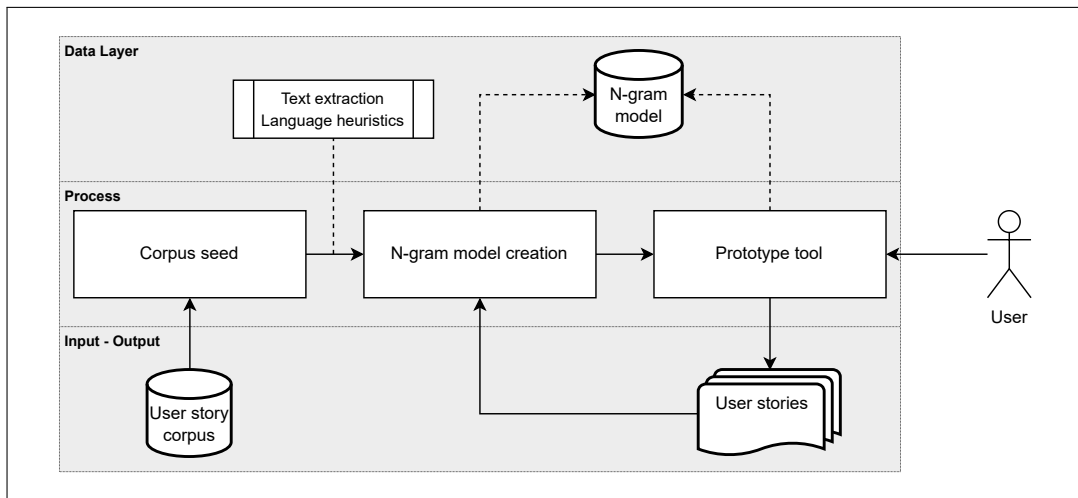


Figure 4.1 : Workflow diagram presenting the creation of the N-gram model and the prototype tool.

We implemented this pipeline with NodeJS and utilized MongoDB for data persistence to facilitate our experiments. We used Docker containers to organize the architecture of our system and ease reproducibility²⁴. We divided the process into two steps: initial training and model adaptation. The first step loads parts of the user story corpus (Dalpiaz *et al.*, 2019) into a database collection through seeding. The second one creates new N-grams while the users save new user stories in the system. All the N-grams are saved into the same MongoDB collection containing the following structure:

$$Ngram_{(collection)} \rightarrow \{input, output, count\}$$

²⁴<https://github.com/betopluga/re-append>

The *input* element is a sequence of tokens representing from 1-gram to 8-gram. The N-gram model was built with $n=9$, an arbitrarily chosen value that proved effective for this statistical approach. It accepts up to eight input tokens and predicts one additional token. The *output* portion is the 1-gram token to be completed based on the previous input. Finally, the *count* represents the number of times the input and output appear on the text. This structure can store all the data we need to implement an N-gram model acting as a collection of n-grams, using short computation time.

INITIAL TRAINING

Initially, to proceed with our start training, parts of the user story corpus (Dalpiaz *et al.*, 2019) were loaded into the database collection through seeding. We extracted the parts of *actor* and the *goal* from the user stories to create some language heuristics. These heuristics are essential to influence our model in following the user story template. Table 4.1 presents the N-grams created using parts extracted from the user story corpus.

The N-gram model proposed can help software practitioners auto-complete parts of the user stories according to the template used. For example, when the user types "As" the model can give two completion options: "a" and "an". After selecting one of the options, the system brings a list of *{actors}* (nouns) extracted from the training corpus used to create the n-grams. The same happens when typing "I want to", but in this case the model brings a list of *{goals}* (verbs) extracted from the same corpus. We also input the 1-gram into our model to ease the typing of some words. It means that when the user types "re" the system can recommend the following completions: "representative", "researcher", "recruiter" and others.

This training step aims to facilitate the typing of recurrent words and reinforce the usage of the user story template. The corpus used to create the N-grams has a vast list of user stories

Table 4.1 : N-grams created based on language heuristics to mimic the user story template.

Type	Input	Output
Template	-	As
Template	As	a
Template	As	an
Template	-	I
Template	I	want
Template	I want	to
Template	-	want
Template	-	to
Template	-	so
Template	-	that
Template	so	that
Actor	As a/an	{actor}
Actor	As a/an {actor}	I want to
Goal / Benefit	-	{goal}
Goal	I want to	{goal}
Goal	want to	{goal}
Goal	to	{goal}
Benefit	so that I can	{goal}
Benefit	that I can	{goal}
Benefit	I can	{goal}
Benefit	can	{goal}

collected from different projects, which means that various words can be used to start writing user stories.

MODEL ADAPTATION

The model is expected to adapt to the project context where the user stories are being written. For this reason, our model can also learn new words when new user stories are stored. We created another MongoDB collection to persist the user stories created by the users. It is a simple structure aimed at storing the text of the user story:

$$US(collection) \rightarrow \{text\}$$

For each user story added to the collection, the N-gram collection is automatically updated by incorporating new tokens and their corresponding frequencies. To accomplish this, we utilized the Natural JS library²⁵, which processes the user stories into n-grams. The new N-grams are stored in the same MongoDB collection as the linguistic heuristics created during the training step. The Algorithm 4.1 presents our pseudo code.

²⁵<https://naturalnode.github.io/natural/>

```

Data: userStories
Result: N-gram model updated
1 CleanNgramModel()
2 foreach userStory ∈ userStories do
3    $n \leftarrow 1$ 
4   foreach  $n \in 1..10$  do
5      $ngrams \leftarrow \text{NaturalJS}(userStory)$ 
6     foreach  $ngram \in ngrams$  do
7        $output \leftarrow ngram(n)$ 
8        $input \leftarrow ngram(1..n - 1)$ 
9       if NGramExist(input, output) then
10        | IncrementNGramCount()
11        end
12        else
13        | CreateNewNgram(input, output)
14        end
15      end
16    end
17 end

```

Algorithm 4.1: N-gram model updating. Line 5 uses the external library **NaturalJS** to extract n-grams from the sentence.

The algorithm proposed also saves all 1-gram tokens into the N-gram model. This means that all words are saved to facilitate typing new user stories. When the N-gram model fails to

predict the next token based on the last ones, the system searches for the 1-gram list to help the user auto-complete the word typing.

SEARCHING TEXT COMPLETIONS

A function was created to search for text completions based on a random *input* string. This function returns all the possible completions sorted descending by *count*. In this way, we ensure that the most probable completions will be listed first. So, given an input string "As a" the model can return the options listed in Table 4.2. It is essential to highlight that this list is dynamic and changes according to the model adaptation process explained before. Another function was coded to list all the 1-gram options to auto-complete words.

Table 4.2 : N-grams found for input search "As a".

Input	Output	Count
As a	representative	2
As a	site member	2
As a	trainer	2
As a	participant	2
As a	data user	1

4.2.1 PROTOTYPE TOOL

We added one more Docker container to our application to create a user interface for our text prediction system. This container has a Vue.JS application containing a simple form for writing and saving user stories using a text completion tool powered by our text prediction N-gram model. This frontend layer connects to our NodeJS backend where we save user stories and storage our N-gram model.

Figure 4.2 is a screenshot of our frontend application presenting the text completion in action. On top of that, we wrote some guidelines to help the experiment participants write their user stories. There is a section behind the form to list all the user stories created in the format of cards. Users can also delete the user stories created.

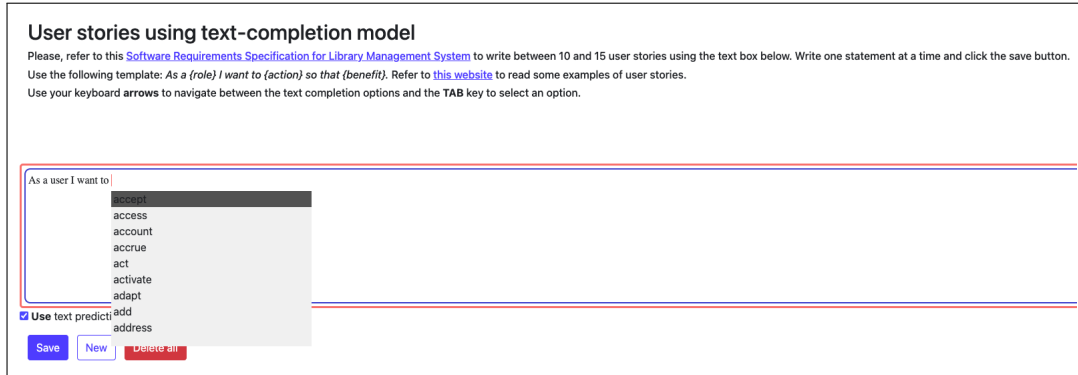


Figure 4.2 : Prototype presenting the assistance tool powered by our N-gram model.

4.3 METHODS

We chose a controlled experiment followed by a survey as an empirical strategy to evaluate our text generation model with software practitioners. A controlled experiment is an empirical investigation in which one factor or variable within the study is deliberately manipulated while keeping others constant. Processes are randomized and applied to the participants, and the effects are measured on the outcome variables (Wohlin *et al.*, 2012b). Figure 4.3 presents the steps of our experiment.

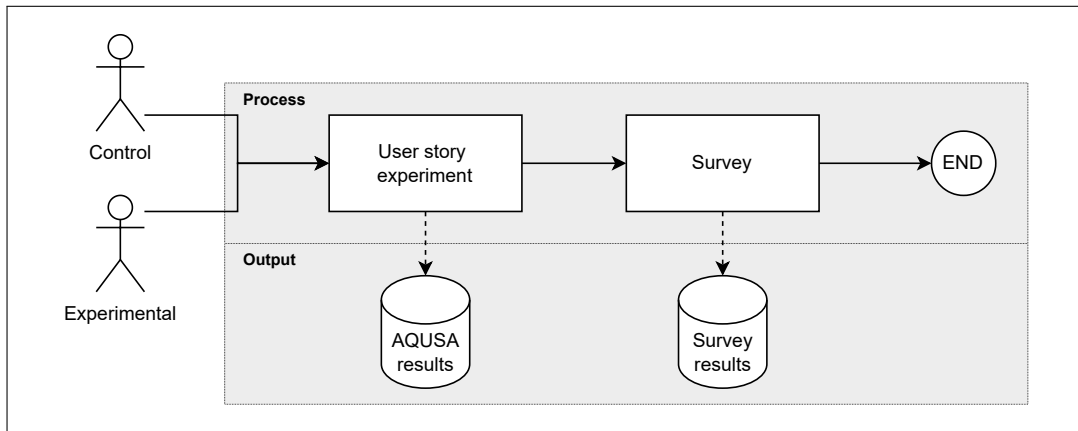


Figure 4.3 : Workflow diagram presenting the steps of our experiment with the expected outcomes.

The scope of our experiment is to identify whether a text prediction system based on the N-gram model can improve the quality of user story statements in the context of ASD using the AQUSA tool as a reference. For this purpose, we planned our experiment as follows (Wohlin *et al.*, 2012b):

4.3.1 INCLUSION CRITERIONS

We conducted our controlled experiment with experienced software engineering professionals to ensure the validity and reliability of the results. To minimize the risk of imbalance, junior professionals were excluded from participation. Instead, we selected software practitioners with at least five years of experience in ASD projects and active involvement in user story practices. As suggested by Falessi *et al.* (2018), the choice of professionals or students as subjects should align with the experiment’s objectives. By involving seasoned practitioners, we aimed to test our text generation model in a more realistic setting, enabling us to gather precise and practical feedback.

Selection of subjects

We performed a *non-probability sampling* technique to select participants for our study. The nearest and most convenient people were chosen as subjects. As described in the *context selection*, we decided to work with professional software practitioners with extensive experience in ASD (over five years). We contacted former coworkers with vast experience in agile software development, including software developers, agile coaches, and business analysts. Those interested in participating received an official email invitation with detailed instructions and a deadline for completing the experiment and the survey. We had 16 participants in our experiment, split equally and randomly into *controlled* and *experimental* groups.

We made our prototype available online on a dedicated server connected to the Internet. All participants were invited to participate in our survey, which allowed them to find a link to the prototype tool. There, they received instructions about writing their user stories using the requirements document of a library system software. They were invited to write ten user stories with or without the assistance of our N-gram model. Ultimately, they returned to the survey to finish participating in the experiment. The study's answers help us draw conclusions about using text generation models to specify user stories.

4.3.2 EXPERIMENT DESIGN

By design, our experiment consists of analyzing one factor with two conditions. The factor is user story specification, and the first condition uses text prediction to assist in writing user stories. The second one is free user story writing without any text assistance. We employed a randomized design, where the participants were randomly and balanced assigned to

each treatment.

Variable selection

This experiment has many *independent variables* that we can control, such as the tool where participants can write user stories, the template chosen, and the requirements document they will use as a reference to write their user stories, among others. We decided to keep all of these variables unchanged.

The only *independent variable* we change in our experiment is the usage of text completion based on the proposed N-gram model. All participants worked on the same website created for this experiment, using the same requirement document for reference, intending to write ten user stories. Our *experimental* group used the tool with the assistance of text completion to write the user stories. In contrast, the *control* group wrote the user stories without any writing assistance.

The *dependent variable* to be assessed derived from our hypothesis is the quality of the user stories resulting. We believe that statements written with our text prediction model have a higher text quality than those written without assistance. This is the object of this study.

Hypothesis formulation

We hypothesize that our N-gram text prediction model improves the quality of user stories written following a determined template. A template enables certain parts of the user stories to be reused, making it easier for our model to recognize these patterns and assist in writing new statements, increasing their consistency and decreasing ambiguity.

4.3.3 VALIDATION

We defined a few requirements to be followed in order to validate the participation of a subject. In addition, we defined the instruments used to evaluate the results.

Instrumentation

We identify three essential instruments to perform our experiment. The first is the prototype tool powered by our N-gram text generation model (presented in Section 4.2.1). Using this object ensures all participants will have the same experience writing user stories, guaranteeing we can control the necessary independent variables of our experiment. Secondly, the survey prepared for both groups is vital to gathering feedback from our prototype tool. Finally, adopting the QUS/AQUSA solution is a measurement instrument to test our hypothesis about writing user stories with text assistance.

[Resketi *et al.* \(2020\)](#) also used the AQUSA ([Lucassen *et al.*, 2015](#)) tool to evaluate and correct user stories before performing their study about reusing. Using AQUSA helped them correct syntactic issues in the user stories and produce a better set of statements for reuse in the future.

Validity Evaluation

We discuss the threats to validity in Section 4.6. Overall, we consider the results obtained from our experiment as they comply with some fundamental requirements:

- Participants should write ten user stories using our prototype. Those who wrote less were invited to complete the experiment. In case they wrote over ten user stories, we applied a cut-off to the last ones to keep all lists in ten.
- The subject participation is only valid if he/she replies to the survey questions at the end of the experimentation.
- We beforehand interview the participants to validate whether they have at least five years of experience working with user stories in ASD.
- All participants must use our prototype to write user stories, even the control group, who have not used the text generation model (we disabled it for them).
- All the user stories should be written in English and follow the user story template proposed.
- We randomly divided participants into controlled and experiment groups, trying to keep the same number of subjects in each group.

4.3.4 SURVEY

We prepared a survey to gather feedback from the experiment participants. After writing user stories with or without text assistance, they replied to the questionnaire. We prepared the questions to allow us to understand the participants' level of acceptance of having automated assistance to write user stories. Writing user stories can sometimes be repetitive, as they use templates and reuse some parts. Automation might help software practitioners write faster and better user stories.

We wrote a set of questions for each group in our experiment. The control group had to reply to the three most general questions, checking how they see the usage of automation in

the area. We also prepared an open question (Q3) to identify the most frequent issues they face when specifying user stories in ASD. Table 4.3 presents the question’s rationale.

Survey questions for the control group		
#	Question	Rationale
Q1	Do you think text auto-completion could help you write better user stories?	This survey question captures the participant’s feelings about whether text generation could help them to write better quality user stories.
Q2	In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, would you use text completion support to write user stories?	This survey question captures the participant’s option about using text generation for user story writing.
Q3	Describe the most frequent issues you face when writing user stories.	The goal of this question is to capture the issues participants faced when writing user stories without any assistance.

Table 4.3 : List of questions and their rationale prepared for the control group.

We defined four questions for the experimental group (participants who used our text prediction model for user stories). These questions aim to identify whether the text prediction could help them write user stories faster (Q1) and better (Q2). We also wanted to know whether they would use text prediction assistance to write user stories in their software development projects (Q3). To close, we repeated our open question of the control group to describe the most frequent issues when writing user stories. Some participants replied on a broader level, while others expressed their experience using our prototype. Table 4.4 presents the question’s rationale.

4.4 RESULTS

We divided this section to present (i) the results of our controlled experiment and (ii) the survey respondents’ responses. The following subsection presents the results collected from the AQUASA tool when evaluating the user stories the subjects wrote. Next, we recap the survey questions and present their answers.

Survey questions for the experimental group		
#	Question	Rationale
Q1	In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, did the text auto-completion help you write faster the user stories?	This survey question investigates whether text generation helped the participants speed up the writing process.
Q2	In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, did the text auto-completion help you write better user stories?	This survey question captures the subject's feelings about whether our generation model has helped them to write better quality user stories.
Q3	In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, would you use text auto-completion to write user stories for your software development projects?	More broadly, we want to know whether the participant would use text generation for user stories in its projects.
Q4	Describe the most frequent issues you face when writing user stories.	The goal of this question is to capture the issues participants faced when writing user stories assisted by our text generation model.

Table 4.4 : List of questions and their rationale prepared for the experimental group.

4.4.1 CONTROLLED EXPERIMENT

Data collected from our controlled experiment consists of the set of user stories written by each participant and the quality results produced by the AQUSA tool ²⁶. We make our results available in a Zenodo repository²⁷.

Using the sets of user stories produced by the participants, we individually applied AQUSA to evaluate the statements written. The result produced by the tool is an HTML file containing all the defects identified in each user story. We used the following command line to run the tool for each set of user stories:

```
pyenv exec python aqusacore.py -i input-userstories.txt
-o output-userstories -f html
```

²⁶<https://github.com/RELabUU/aqusa-core>

²⁷<https://doi.org/10.5281/zenodo.14639834>

As discussed, this tool can only evaluate syntactic and pragmatic quality criteria, not semantic attributes. However, this is enough for the level of analysis we want to cover in our study. Table 4.5 presents the results produced by the AQUUSA tool.

AQUUSA Results				
Category	Defect kind	Subkind	Experimental	Control
Syntactic	Atomic	Conjunctions	7	22
Pragmatic	Uniform	Uniform	4	6
Syntactic	Minimal	Indication repetition	0	2
Syntactic	Minimal	Punctuation	0	3
Syntactic	Minimal	Brackets	0	1
Total			11	34

Table 4.5 : Results after running the AQUUSA tool in the user stories produced by both study groups: experimental and control. The tool evaluated all syntactic criteria listed in Table 1.2 and part of the pragmatic attributes (*explicit dependencies, uniform and unique*).

Specifically, 75.6% of the defects identified originated from the control group, creating user stories without text generation assistance. In contrast, only 24.4% of the defects were associated with the experimental group, where user stories were written with the support of our text generation model. Furthermore, the control group exhibited a higher number of defects in both *syntactic* and *pragmatic* categories. These results suggest that leveraging our text prediction model significantly improved the quality of user stories in our controlled experiment.

Our attention was drawn to the fact that the experimental group did not produce any defects classified as *minimal* (sub kinds *indication repetition, punctuation and brackets*), while the control group produced six defects of these sub kinds. Additionally, the control group had a significantly higher number of user stories that failed to be *atomic* (22 user stories) compared to the experimental group, which had only seven. Lastly, the experimental group demonstrated a slight advantage in producing more uniform user stories: only four user stories deviated from the template, compared to six from the control group.

EXAMPLES OF DEFECTS

We opted to present some examples of the defects found by the AQUASA tool in the user stories the participants wrote. It can give a better idea of how text prediction could assist the participants in improving the quality of the text.

US01: *As a Library Manager, I want to see an interface that allows me to remove, change and add user accounts and account information so I can easily manage the library staff.*

The user story above has a defect of kind *atomic* and sub kind *conjunctions*. Conjunctions often lead to non-atomic user stories by introducing multiple requirements, conditions, or actions within a single story. It makes splitting, tracking, and prioritizing the development work harder.

US02: *As a Library Manager, I want to generate a report of all sign out books, so I can manage the librarys collection more effectively.*

The previous user story has a defect of kind *uniform* and sub kind *uniform*. In other words, the participant failed to follow the user story template (As a/an, I want to, so that). In this case, the sentence lacks the word *that*. The expression "so" is a response or consequence, while "so that" introduces a subordinate clause that shows an intended result or effect ([Adam, 2015](#)). Formally, they have different meanings that lead the user story to not comply with the template recommended.

US03: *As a library staff member I want to generate a report showing the information about all the users who have overdue books and penalty so what I can have a complete overview of all borrowed books and fees as well as perform recalls to these users.*

The previous user story has a defect of kind *minimal* and sub kind *indication repetition*. Indicator repetition points to opportunities for improving the story's conciseness and focus. By eliminating redundancy, the story becomes more precise and more atomic. In this case, the sentence redundantly emphasizes "information," "overdue books," and "penalty," which could be streamlined.

US04: *As a Library Manager, I want to update a book record with the following information: category, title, ISBN, publisher, book description, location, purchase date and price, so that this book information will be available for the library system interface.*

AQUSA points a *minimal* defect of sub kind *punctuation* to any user story that contains additional text after a dot, hyphen, semicolon, or other separating punctuation marks. In the US04, the participant used a colon to list some items. It violates the user story's template and makes the user story less atomic.

US05: *As a Library Manager, I want to print a report so that all the information displayed on the report will be available outside the system (either printed or file storage).*

To close, AQUSA also reports user stories containing *brackets* as a *minimal* defect. In the last user story, the participant used brackets at the end of the sentence. This practice can increase the ambiguity of the sentence and sometimes indicate the presence of more requirements.

4.4.2 SURVEY - CONTROL GROUP

In this section, we present the answers of the control group to our post-experiment survey:

Q1 - Do you think text auto-completion could help you write better user stories?

All respondents agree that text prediction could help them write better user stories (8/8). Nowadays, various text generation applications are being used for different purposes. Users already know the advantages of using them and might notice some improvement in writing quality when supported by these tools. This perception leads software practitioners to consider using text generation for their projects as a goal to save time and improve the quality of requirement statements.

Q2 - In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, would you use text completion support to write user stories?

The participants would strongly like to use text prediction to write their user stories (9.38/10). We expected that text prediction would be widely accepted for this purpose. Software practitioners usually write user stories during refinement meetings and customer conversations (Cohn, 2004), so any resource that could help them support this process seems welcome.

Q3 - Describe the most frequent issues you face when writing user stories.

As this is an open question, the participants are expected to share various opinions about the issues faced when writing user stories. Some answers are straightforward, while others are more discursive. We categorized the answers into groups to summarize our results and ease readability. Table 4.7 presents the issues reported by respondents.

Notably, the most reported issue is difficulty comprehending and translating business rules into user stories (6/8). *Pc1* and *Pc2* reported to be a challenge to identify the value added for each user story and write it on the benefit portion (*so that*). *Pc1* and *Pc3* report

Control Group Survey Results		
Participant	Q01	Q02
Pc1	Yes	8
Pc2	Yes	10
Pc3	Yes	10
Pc4	Yes	9
Pc5	Yes	9
Pc6	Yes	10
Pc7	Yes	9
Pc8	Yes	10
Average		9.38

Table 4.6 : Summary of the control group survey results of 8 participants. Q3 is an open question asking participants to summarize the main challenges when writing user stories.

Issues reported by the control group	
Issue	Participants
Repetitive task	Pc5, Pc7, Pc8
Comprehend/express business rules	Pc1, Pc2, Pc3, Pc4, Pc6, Pc7
Dependency management	Pc1, Pc6
Grammar/spelling	Pc2
Writing	Pc1, Pc3, Pc6, Pc8
Time-consuming	Pc1, Pc6

Table 4.7 : Issues reported in Q3 by the control group.

that it is hard to define all the cases and corner cases in user stories and make it easier for the development team to understand. *Pc6* also believes that it is not easy to summarize some business rules into short descriptions. *Pc4* was straightforward, saying writing user stories is challenging when the business rules are unknown. Finally, *Pc7* finds issues when writing and organizing user stories for different types of users.

Writing was the second most reported issue, slightly connected to the first. We understand that some participants struggle to write user stories because they cannot translate the business rules into short descriptions. *Pc1*, *Pc3*, and *Pc6* concluded that it is challenging to

keep user stories clear and concise to avoid multiple interpretations. *Pc1* mentions that it is hard to balance complexity and generality. Complex user stories could be hard for the team to understand and implement, and generic ones could be too vague.

Pc1 and *Pc8* believe the lack of modularity is a problem. It is easy to understand who authored each user story by the writing style. It indicates some teams might fail to follow a template. *Pc2* also cited that he usually faces grammar and spelling issues when writing user stories without any text completion or correction tool assistance.

Participants also reported writing user stories as repetitive. *Pc8* highlights that user stories for CRUD operations tend to repeat some parts, keeping the same structure but slightly changing the acceptance criteria and the expected results. *Pc5* says that as user stories follow templates, their parts could be reused to automatically auto-complete others. On the same line, some participants mentioned that writing user stories is a time-consuming task (*Pc1* and *Pc6*).

Lastly, some respondents also mentioned the challenge of user story dependency and management. *Pc1* finds it challenging to establish the connection and interference among the project user stories. *Pc6* mentioned that it is hard to link them with correlates.

4.4.3 SURVEY - EXPERIMENTAL GROUP

Q1 - In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, did the text auto-completion help you write faster the user stories?

The text completion assisted the participants in writing user stories faster (7.5/10). In a quick post-experiment interview with participants, they found it handy to have the text assistant complete some parts of the user stories, mainly when writing the last ones, when the

Experimental Group Survey Results					
Participant	Q01	Q02	Q03	Q4 (General)	Q4 (Experiment)
Pe1	9	9	10		x
Pe2	4	4	4		x
Pe3	8	3	10	x	
Pe4	9	8	10	x	
Pe5	8	9	10		x
Pe6	7	6	8	x	
Pe7	8	7	10	x	x
Pe8	7	8	10		x
Average	7.5	6.75	9		

Table 4.8 : Summary of survey results of the experimental group of 8 participants, capturing responses to 4 questions as part of the study’s evaluation process. Q4 is an open question asking participants to summarize the main challenges when writing user stories. Some participants replied to the Q4 more broadly, and others restricted to the experiment.

model could learn terms from the first ones created.

Q2 - In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, did the text auto-completion help you write better user stories?

The participants believe that the auto-completion of the text neither improved nor harmed the quality of the user stories written (6.75/10). *Pe3* gave the lowest score to this question (3) while *Pe1* and *Pe5* gave the highest ones (9). We can be inclined to say that the participants believe the text completion could somehow improve the quality of the user stories. However, it depends on the personal experience of each respondent, as their capacity to evaluate how they would be writing the same stories without text assistance.

In summary, although it counts in our analysis, we believe it is a subjective opinion that needs to be better evaluated using more strict guidelines. That is why we employed the AQUSA tool to evaluate user stories. Combining the results collected from AQUSA and the

answer to this question might help us guide our discussion about using text generation to improve the quality of user stories.

Q3 - In a range of 1 to 10, being 1 = not at all likely and 10 = extremely likely, would you use text auto-completion to write user stories for your software development projects?

The participants likely would use text auto-completion to write user stories for their software development projects (9/10). After being part of our experimentation, the participants tend to use text-completion to write user stories. In general, we conclude that this idea is well-accepted by software practitioners. We mention the score for the same question applied to the control group as evidence: 9.38.

Q4 - Describe the most frequent issues you face when writing user stories.

Some experimental group participants replied more broadly to this open question, while others were more specific about their experience using our prototype. Table 4.9 summarizes the answers into categories.

Pe3 shared that the main challenge when writing user stories is to make them short and clear enough for the development team to understand. Also, writing the acceptance criteria is laborious. In the same way, *Pe6* said it is hard to ensure the tech team understands the requirements.

Pe4 pointed out that it is hard to update user stories as they evolve as the project progresses. In addition, he cares about the quality of the requirements received from the stakeholders to specify the correct user stories.

Issues reported by the experimental group	
Issue	Participants
Concise and clear statements	Pe3, Pe6
Write acceptance criteria	Pe3
Track changes	Pe4
Defective requirements	Pe4
Lacking user perspective	Pe7
Lacking context*	Pe1, Pe7, Pe8
User experience*	Pe2, Pe7

Table 4.9 : Issues reported by the experimental group related to Q4. Some participants replied to the question more broadly, while others shared their experience using our prototype. *The last two issues are related to our prototype.

Pe7 contributed to our survey, stating that Product Owners (POs) frequently don't write stories using the user perspective. Many POs write user stories using their viewpoint or the team's perspective. The participant concluded that it would be good to have a predictive system to help the development team write user stories from the user's perspective.

Pe1, Pe2, Pe7 and *Pe8* shared their experience using our prototype. As said, the goal of the open question Q4 was to capture the challenges the participants face when writing user stories in real scenarios. However, we are happy some participants shared their thoughts about our experimentation tool.

Pe1, Pe7 and *Pe8* claimed the lack of context when writing user stories referencing the requirements document we shared. We did not use this document to train our predictive model. Instead, we used a corpus of user stories as explained in Section 4.2. We agreed because if we had used the document as part of our corpus, we might have had better text completions.

To finish, *Pe2* and *Pe7* believe that the user experience of our prototype was poor. We agree, and if we perform another experiment, we will address the necessary improvements to our tool.

4.5 DISCUSSION

The feedback from the participants helped us validate our N-gram model and our research hypothesis. The experimentation and survey aimed to test whether text prediction could help software practitioners write *faster* and *better* user stories.

We initially trained our text generation model with a corpus of random user stories from actual software development projects. This initial model assisted the participants in writing their first user stories using the suggestions based on the template of user stories (*actor*, a *goal*, and the *benefit*). While the participants were using the prototype, the N-gram model was updated with new prompts and outputs based on the N-grams extracted from the user stories saved. After some user stories had been written, they noticed that our text completion suggested N-grams from previous user stories (*Pe1* and *Pe5* reported it).

We concluded that text prediction can assist in writing and improve the quality of user stories. Around 75% of the defects found were in the control group, while others 25% came from the experimental group, which used our text generation approach to auto-complete user stories; neither defect of category *syntactic (minimal)* was found in the user stories written by the experimental group, and even the *syntactic* and *pragmatic* defects found were less present in this group.

Analyzing the user stories created and the results reported by the AQUASA tool (some examples in Section 4.4.1), we concluded that our approach improved the uniformity of the user stories. *Uniformity* is reached when the format of the requirements statements follows a standard to minimize miscommunication and facilitate project management. This is one of the quality attributes of the *Agile Requirements Verification Framework* proposed by [Heck & Zaidman \(2014\)](#). How our N-gram model was built encourages the writer to follow the user story template.

Analyzing the feedback provided by some participants, we also found that our approach could improve the *consistency* of the requirements as defined in the ISO/IEC/IEEE 29148 (ISO, 2018). According to the standard, consistency means that the set of requirements contains individual requirements that are unique, without conflicts or overlaps. Also, the terminology used within the set of requirements is consistent; in other words, the same term is used throughout different statements to mean the same thing. As our N-gram model stores the terms used in all user stories saved, it helped the participants reuse them in different user stories, increasing the consistency among the statements; inconsistency was mentioned by Amna & Poels (2022a) as an under-researched issue in user story practice.

Having more *consistent* and *uniform* user stories reduces ambiguity and eases communication between team members and stakeholders. We employed a statistical N-gram model with structured heuristics to guide the writer in following a template and reusing terms already saved in previous user stories. This model could be pre-trained with a more specific corpus to better adhere to the context (as suggested by some participants). In a real scenario, we could use agile requirements artifacts (documents, e-mails, meeting transcriptions), particularly for the project we need to write user stories. In such circumstances, we could reuse N-grams of the requirements statements to auto-complete the user stories following the *Connextra* template.

Other improvements could be applied to the N-gram model to give better text predictions to auto-complete user stories. We understand that it is another step in our research. We might refine our technique or even employ LLMs (*Large Language Models*) to look for the best text prediction model for user stories. LLMs also have limitations and challenges (Laskar *et al.*, 2024; Raschka, 2024) despite being State-of-the-Art for many NLP tasks. For now, we confirmed through our experiment that our approach could assist in writing and improving the quality of the user stories.

4.5.1 SURVEY

The *Q1* and *Q2* prepared for the control group, and *Q3* of the experimental one aimed to validate the interest of software practitioners in automated support for user story writing. The responses essentially confirmed the acceptance of participants by the topic. It can incentivize future research on the subject.

Q1 prepared for the experimental group aimed to understand whether our text prediction model helped the participants write the user stories *faster*. Their answers confirmed our assumption (7.5/10). Secondly, *Q2* confirmed with a grade 6.75/10 that our approach could help participants to write user stories *better*. This question aimed to capture the participant's feelings about the quality of the user story written. The final grade of this question is a bit ambiguous. We cannot affirm that our text-prediction model harmed the user stories nor support all participants as expected. We concluded that some slight changes to our model could help software practitioners have a better experience with our prototype.

The participant's responses to the open questions from both groups revealed interesting insights about the employment of user stories in ASD. According to the control group, the most reported issue faced when writing user stories is *comprehending and expressing business rules*. Employing user stories and agile practices brings development teams closer to the business, supporting requirements refinement and contacting stakeholders. Dealing with business rules can be challenging for team members because they lack domain knowledge. It can be mitigated by the participation of domain experts and users in requirements refinement meetings (Cohn, 2004).

We also noticed that some participants claimed that *writing user stories is repetitive*. As the user stories follow a template, it is standard that some parts are reused. This was one of our motivations when preparing this research. Based on the authors' practical experience, we

found that a text predictive assistant could speed up the writing and improve the quality of the user stories.

Moving again to the experimental group, the most cited issue by participants is related to the *lack of context* of our N-gram model. As explained before, it happens because we did not use the requirements document used by participants to write user stories as a corpus to train our model. So, it was not expected to reuse the terms in the user stories. Also, participants reported issues with the user experience of our prototype. We agree that we could address some improvements to our prototype.

Some participants reported that *writing concise and clear user stories* is challenging. Sometimes, it can be hard to translate requirements into user stories because requirements can be complex, while user stories are short descriptions that guide future conversations. Analyzing our results, we believe using text prediction can mitigate it because it will guide the user story writing following templates or good practices.

4.5.2 LIMITATIONS

While text generation techniques have shown promise in improving the consistency and structure of user stories, there are inherent limitations to their application. One significant challenge lies in capturing the full nuance of user intent and domain-specific knowledge. User stories often contain implicit assumptions, contextual references, or stakeholder-specific language (Cohn, 2004) that automated systems may struggle to interpret or reproduce accurately.

Without proper context, generated user stories can risk being overly generic, lacking in actionable detail, or misaligned with actual business goals. Furthermore, current models may struggle to represent edge cases or evolving requirements that are common in real-world software projects. These challenges are compounded by the fact that ambiguity in user stories

is not only a known problem but also one that lacks a comprehensive solution. As identified by [Amna & Poels \(2022a\)](#), ambiguity often arises from human behaviors and cognitive factors, and is rarely studied as a property of a set of related user stories, such as themes or epics. Additionally, existing approaches tend to overlook the complexity of ambiguity as it spans multiple linguistic levels. In this context, text generation techniques must be carefully evaluated and complemented by human oversight to avoid introducing or amplifying ambiguities across interconnected user stories.

Another limitation is the dependency on the quality and diversity of training data. Text generation models trained on limited or homogeneous datasets may reinforce existing patterns or introduce biases, leading to repetitive or unrealistic outputs ([Bender & Friedman, 2018](#)). Additionally, while templates and language models can promote consistency, they may also reduce flexibility and creativity, especially when teams need to express non-standard scenarios or innovative features. Ultimately, text generation should be seen as a support tool rather than a replacement for human judgment. Human oversight remains essential to ensure that user stories are relevant, feasible, and accurately reflect user needs and business priorities.

4.6 THREATS TO VALIDITY

We used [Campbell & Stanley \(2015\)](#) framework of threats to validity, selecting only the threats applicable to our controlled experiment.

Internal Validity.

- *Selection:* The participants in the control and experimental groups might differ in their skills, experience, or familiarity with user story writing. This could lead to differences in performance unrelated to the text prediction model. *Mitigation:* Participants were

randomly assigned to groups, ensuring a balance in their experience levels and familiarity with user story writing practices. Also, we ensured that all participants had at least five years of experience working with user stories in ASD.

- *Maturation*: Participants' performance might improve during the experiment due to practice with the task. *Mitigation*: We asked them to write only ten user stories to keep the experiment short and mature.
- *Testing Effects*: Prior exposure to similar tasks or instructions could influence participants' performance. *Mitigation*: Neither participant received a pre-test or similar task before the experiment started.
- *Instrumentation*: Variability in how the user stories were evaluated, or inconsistencies in the provided instructions could affect the outcomes. *Mitigation*: We used the same evaluation method for all user stories written: issues report generated by the AQUASA tool.
- *Experimenter Bias*: Researchers' expectations might unintentionally influence how participants interact with the tool or how their output is evaluated. *Mitigation*: We did not mention any expectation about using text prediction to user stories, so the participants could not have pre-expectations.

External Validity.

- *Population Validity*: The experiment was conducted with a specific group of software practitioners, which may not fully represent the diversity of the software development industry. *Mitigation*: We included diverse participants with varying experience levels and backgrounds and from different companies

- *Ecological Validity*: The controlled experimental setting may not reflect real-world conditions where user stories are often written collaboratively, under time constraints, and integrated into broader workflows. *Mitigation*: While the experiment simulated some realistic conditions, the limitations of the controlled environment are noted.
- *Tool Familiarity*: Participants in the experimental group might have been limited by their unfamiliarity with the text prediction feature, affecting their performance. *Mitigation*: We wrote a short manual advising how to use our tool before the text field where participants wrote the user stories.
- *Task Representativeness*: The software requirements document used in the experiment might not fully represent the diversity and complexity of real-world requirements. *Mitigation*: The document included a mix of typical and complex requirements. Some participants lacked more diagrams and business rules to write the *so that* portion of the user stories.

4.7 CONCLUSION

This chapter investigated the impact of automated text prediction on the quality of user stories written by software practitioners. We prepared a N-gram model with a corpus of user stories extracted from real software projects and set up heuristics to guide the writing using the *Connextra* template. This model has been made available through a web prototype tool for validation.

We prepared a controlled experiment with 16 participants split into control and experimental groups. The first group used the prototype to write user stories without text prediction, while the second used the model. We found that using our text prediction model enhanced the *consistency* and *uniformity* of the user stories compared to the manual writing method.

We also prepared a post-experiment survey for each experiment group to reply to. This survey helped us to confirm the acceptability of text generation models to assist user story writing. In addition, we collected feedback from the participants about their experience using the proposed model. We asked them whether the model helped them write better user stories faster. These findings underscore the potential of leveraging natural language processing (NLP) techniques to address challenges in agile requirements engineering, particularly in user stories.

The research has practical implications, including the potential for text prediction tools to support the requirements specification process, reduce ambiguity, and enhance team productivity in ASD. As these tools evolve, their role in facilitating more effective communication and collaboration within software teams will become increasingly important.

For future work, we plan to refine our text generation model for user stories to improve its accuracy and our prototype tool's usability. Also, we plan to redo the experiment with new participants using the latest version of our model. In this sprint, we might opt for students or junior professionals as a goal to investigate whether text prediction could support them in writing better user stories.

CONCLUSION

This thesis explores whether text prediction assistance helps software practitioners write user stories *faster* and *better*. User stories play a pivotal role in Agile Software Development (ASD), assisting development teams to build the right software product (Lucassen *et al.*, 2016b). They are short statements centred on the user's perspective about the system, are easy to write and comprehend, and act as reminders for future requirements refinement. Software development teams and stakeholders rely on these statements to reinforce communication and plan development activities in Software Engineering (SE) (Schön *et al.*, 2017; Lucassen *et al.*, 2016b).

As per their essence, user stories are prone to ambiguity, affecting requirements elicitation and specification when poorly written (Amna & Poels, 2022a). The variety of data sources and textual formats used as input to Requirements Engineering (RE) poses a challenge to software development teams (Aranda *et al.*, 2010). They are responsible for understanding the client's needs from various sources and translating them into user stories or other requirement specification formats. This process is typically performed manually during refinement meetings, requiring considerable time and effort from the team during each software development iteration.

Throughout the chapters of this thesis, we employed text generation approaches to generate user stories automatically. We hypothesized that text generation could assist software practitioners in formulating better user stories. We concluded that the generation models sped up the writing and created a more consistent and uniform set of statements, reducing ambiguity and improving communication among team members. Although in its infancy, the topic draws the attention of software practitioners and has the potential to consolidate as a research area in SE.

CONTRIBUTION

In Chapter 1, we presented the theoretical background about Agile Software Development ASD, Requirements Engineering (RE), Artificial Intelligence (AI) and text prediction approaches and evaluation methods. This solid knowledge is the background to the following chapters, where we employed different approaches to develop text generation models to assist the writing of user stories.

In Chapter 2, we presented our SLR about automatic user stories generation in the context of SE, presenting the state-of-the-art in the topic. This study ([dos Santos et al., 2024](#)) was published in the *International Journal of Data Science and Analytics* in June 2024, entitled *Automatic user story generation: a comprehensive systematic literature review*.

This study concluded that:

- There is a scarcity of user stories corpora publicly available for training text generation models, hindering advancements in this field, especially in the current era of ML. The authors utilized datasets of varied formats and sources, which proved instrumental in generating user stories. This versatility can be attributed to the inherent nature of user stories, which often stem from diverse sources across different stages of software projects. The identified datasets publically available are [Meetings Transcripts](#), [Software Issue Tracker CONNECT](#), [Requirements dataset \(user stories\)](#), [PURE](#) and [US/BDD scenarios](#).
- A wide variety of Artificial Intelligence (AI) techniques are being employed to support the generation of user stories. Natural language preprocessing techniques, such as POS tagging and text parsing, are familiar ground among the studies and effectively

support the classification of user story data. Generating user story statements is mostly a combination of language heuristics and natural language processing techniques.

- To finish, the studies need to pay more attention to guidelines for evaluating the quality of the user stories generated. Authors could use more qualitative measurements to ensure software practitioners can use the generated user stories in project scenarios. Only two studies ([Lam et al., 2022](#); [Resketi et al., 2020](#)) employed a more qualitative measurement of the user stories produced by their approaches.

This study's findings helped us guide the following steps of our research. One of the user stories corpora identified in our SLR was used to train our text generation models. The significant employment of NLP techniques motivated us to avoid more complex machine learning algorithms initially and rely on linguistic heuristics and common ground NLP techniques. We planned to shift to other algorithms if we could not reach our desired results with pure NLP. Finally, the lack of qualitative measurement of the user stories generated motivated us to look for more strict evaluation guidelines for user stories. We employed the AQUASA tool supported by a post-experiment survey questionnaire to evaluate our text generation model (used in Chapter 4).

After, in Chapter 3, we presented two distinct approaches to create text generation models using N-grams extracted from a user stories dataset. This study ([Dos Santos et al., 2024](#)) was published in the *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)* in February 2024, entitled *AI-Driven User Story Generation*.

We automatically generated user stories using two distinct approaches: N-gram representation with linguistic heuristics and the GPT-3 model. We created a dictionary of N-grams extracted from a user story dataset for the first model. We combined them with linguistic

heuristics to follow the user story *Connextra* template. We fine-tuned a GPT-3 model for the second model using the N-grams stored in our dictionary. We simulated the generation of user stories using both models, and we evaluated them by calculating the metrics BLEU, ROUGE-N, and BERTScore for each set of user stories generated by both approaches. For the N-gram approach, we achieved an average: ROUGE-N=0.39, BLEU=0.26, BERTScore=0.73; and for the GPT-3 model: ROUGE-N=0.46, BLEU=0.27 and BERTScore=0.69. We concluded that although GPT models excel in producing more comprehensive user stories, N-gram models exhibit higher semantic sensitivity. Considering its simplicity and minimal processing requirements, we recommend employing the N-gram technique for user story generation.

The achievements of this study motivated us to continue working with the N-gram approach supported by linguistic heuristics. We refined our approach and performed a controlled experiment followed by a post-experiment survey to evaluate the user stories written by participants using stricter guidelines for user story evaluation. We present this new study in the Chapter 4. We submitted this study as a journal paper to the *Requirements Engineering Journal* in January 2025.

In the mentioned study, we refined our text prediction model to assist in drafting user stories, aiming to reduce writing errors and accelerate the specification process. We prepared a controlled experiment with sixteen participants split into experimental and control groups. Every group was invited to write user stories using a sample of software requirements as a reference. The first group utilized our text prediction model to auto-complete sentences, while the second group created user stories independently. To assess the quality of the user stories, we employed the AQUASA tool, which evaluates both syntactic and pragmatic aspects.

Our analysis revealed that 75.6% of the defects identified were in the control group, compared to only 24.4% in the experimental group. We found that using our text prediction

model enhanced the *consistency* and *uniformity* of the user stories compared to the manual writing method. By definition, consistency means that standard terminology is used within the set of requirements (ISO, 2018). Our N-gram model stores the terms used in all saved user stories, which helps to reuse them in new user stories, increasing the consistency among the statements. Also, uniformity is reached when the format of the requirements statements follows a standard to minimize miscommunication and facilitate project management. The N-gram model produced was trained to follow the *Connextra* template to user stories, reinforcing the uniformity of the statements.

To close, we conducted a post-experiment survey to gather participant feedback. The survey results confirmed that software practitioners are interested in adopting text generation for user stories. The experimental group agreed that our model sped up the writing process and partially agreed that it improved the quality of their user stories. Finally, some participants shared important insights about their experience using user stories. These insights can help guide future research on the topic.

LIMITATIONS

Despite the achievements presented before, this study also has some limitations. One notable limitation is the scope of the controlled experiment (Chapter 4), which was conducted in a simulated environment with a specific set of parameters. While this approach allowed for detailed analysis and consistency, it may not fully capture the variability and complexity of real-world scenarios. Additionally, the selection of participants, although representative of the target population, was limited in size and diversity, which could impact the applicability of the findings.

Another limitation lies in the technical and contextual scope of the study. This research focused predominantly on adopting the *Connextra* template to write user stories. However,

different user story templates are used in the industry, and not all companies rely on templates. This poses a challenge when creating text prediction models to support user story composition. Given this scenario, the scope of this research could be increased to ensure broader applicability in the software industry.

FUTURE WORK

Adopting text prediction models for user story composition is a novel topic in SE and has recently drawn attention from researchers ([Marques *et al.*, 2024](#); [Borg, 2024](#); [Arora *et al.*, 2024](#); [Ronanki *et al.*, 2023a](#); [Zhang *et al.*, 2023](#)). This research collaborates by introducing the concept of user story quality improvement through the employment of text prediction. As presented before, this research also brings limitations that could be addressed by future work.

This study's focus on only one user story template is a known limitation. In future research, we plan to consider other templates or free-style writing due to the dynamic use of user stories ([Wagner *et al.*, 2019](#)). This would address changes in our approach to creating text prediction models, which should adapt to different writing standards. In this sense, we might improve our N-gram approach presented in Chapters 3 and 4 by adding more linguistic heuristics or start exploring more about Large Language Models (LLMs) to add more creativity in the text predictions.

A range of open source LLMs could be used to train a model for user story composition. For example, we could fine-tune or train LLaMA ([Touvron *et al.*, 2023](#)) LLM for this task. Meta Platforms owns LLaMA. The model was introduced in February 2023 and designed to be more efficient and accessible for researchers than other LLMs.

Creating new text prediction models and addressing different use story templates will result in new controlled experiment setups. The adoption of the AQUASA tool is limited to the

Connextra template (Lucassen *et al.*, 2016a). Therefore, evaluating the acceptability of new models through experimentation is essential to ensuring the models will facilitate composition and improve the writing quality of user stories. We will have to adjust the evaluation protocol to check the quality of the user stories, maybe asking participants to make part of this process.

Ultimately, we could explore the ethical implications of using text generation models in the creation of user stories, especially as these tools become more integrated into software development workflows. Key concerns include the transparency of the content generated by artificial intelligence tools, the potential reinforcement of bias from training data, and the erosion of stakeholder accountability when requirements are produced automatically. Research is needed to establish guidelines and practices that ensure generated stories are traceable, inclusive, and aligned with real user needs.

Notably, although the present studies helped us confirm our thesis hypothesis, this work is still ongoing. The search for a text prediction model for all-cases scenario user story composition has just started. Many variables still need to be addressed to consider the broader applicability of our idea; thus, we prefer to base ourselves on the quote of *Lao Tzu*: "*The journey of a thousand miles begins with a single step*". The first step has been taken.

REFERENCES

(2018). Peters, Matthew E. and Neumann, Mark and Iyyer, Mohit and Gardner, Matt and Clark, Christopher and Lee, Kenton and Zettlemoyer, Luke.

(2024). *What Is a Chatbot?* Retrieved 2023-04-06, from <https://www.ibm.com/topics/chatbots>

Adam (2015). *Learn English Grammar: How to use SO and SO THAT*. Retrieved 2025-01-04, from <https://www.engvid.com/english-grammar-so-that/>

Amna, A. R. & Poels, G. (2022). Ambiguity in user stories: A systematic literature review. *Inf. Softw. Technol.*, *145*, 106824.

Amna, A. R. & Poels, G. (2022). Systematic Literature Mapping of User Story Research. *IEEE Access*, *10*, 51723–51746.

Aranda, G. N., Vizcaíno, A. & Piattini, M. (2010). A framework to improve communication during the requirements elicitation process in GSD projects. *Requir. Eng.*, *15*(4), 397–417.

Arora, C., Grundy, J. & Abdelrazek, M. (2024). Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs (129–148).

Aurum, A. & Wohlin, C. (2005). *Engineering and Managing Software Requirements*. Springer.

Beck, K. & Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. XP Series. Pearson Education.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). *Manifesto for Agile Software Development*. Retrieved 2023-04-30, from <http://www.agilemanifesto.org/>

Bender, E. M. & Friedman, B. (2018). Data Statements for Natural Language Process-

ing: Toward Mitigating System Bias and Enabling Better Science. *Transactions of the Association for Computational Linguistics*, 587–604. doi: [10.1162/tac1_a_00041](https://doi.org/10.1162/tac1_a_00041)

Bird, S., Klein, E. & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. Sebastopol, USA: O'Reilly Media, Inc.

Bokka, K., Hora, S., Jain, T. & Wambugu, M. (2019). *Deep Learning for Natural Language Processing: Solve your natural language processing problems with smart deep neural networks*. Packt Publishing.

Borg, M. (2024). Requirements Engineering and Large Language Models: Insights From a Panel. *IEEE Software*, 41(2), 6–10.

Bourque, P. & Fairley, R. E. (dir.) (2014). *SWEBOK: Guide to the Software Engineering Body of Knowledge* (version 3.0). Los Alamitos, CA: IEEE Computer Society. Retrieved 2023-04-30, from <http://www.swebok.org/>

Buddana, H., Kaushik, S., Manogna, P. & P.s., S. (2021). Word Level LSTM and Recurrent Neural Network for Automatic Text Generation. *2021 International Conference on Computer Communication and Informatics, ICCCI 2021*.

Campbell, D. & Stanley, J. (2015). *Experimental and Quasi-Experimental Designs for Research*. Houghton Mifflin Company, Boston, USA: Ravenio Books.

Castillo-Barrera, F. E., Amador-García, M., Pérez-González, H. G., Martínez-Pérez, F. E. & Torres-Reyes, F. J. (2018). Adapting Bloom's Taxonomy for an Agile Classification of the Complexity of the User Stories in SCRUM. In *2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 139–145. IEEE.

Cheligeer, C., Huang, J., Wu, G., Bhuiyan, N., Xu, Y. & Zeng, Y. (2022). Machine learning in requirements elicitation: a literature review. *Artif. Intell. Eng. Des. Anal. Manuf.*, 36, e32.

Chopra, R. (2018). *Software Quality Assurance: A Self-Teaching Introduction*. Mercury Learning & Information.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S. & Fiedel, N. (2022). PaLM: Scaling Language Modeling with Pathways.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. USA: Addison Wesley Longman Publishing Co., Inc.

Cohn, M. (2009). *Succeeding with Agile: Software Development Using Scrum* (1st). Addison-Wesley Professional.

Cohn, M. (2019). *What is a user story template and why does it work so well?* Retrieved 2023-04-25, from <http://www.agilemanifesto.org/>

Cruzes, D. S. & Dybå, T. (2011). Recommended Steps for Thematic Synthesis in Software Engineering. In *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*, 275–284. IEEE Computer Society.

Dalpiaz, F., van der Schalk, I., Brinkkemper, S., Aydemir, F. B. & Lucassen, G. (2019). Detecting terminological ambiguity in user stories: Tool and experimentation. *Information and Software Technology*, 110, 3–16.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

dos Santos, C. A., Bouchard, K. & Minetto Napoleão, B. (2024). Automatic user story generation: a comprehensive systematic literature review. *International Journal of Data Science and Analytics*.

Dos Santos, C. A., Bouchard, K. & Petrillo, F. (2024). AI-Driven User Story Generation.

In *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, 1–6.

dos Santos, C. A. & Petrillo, F. (2021). Towards auto-completion on software requirements statements.

Dwitam, F. & Rusli, A. (2020). User stories collection via interactive chatbot to support requirements gathering. *Telkomnika (Telecommunication Computing Electronics and Control)*, 18(2), 890–898.

Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A. & Oivo, M. (2018, 1). Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Softw. Engg.*, p. 452–489.

Fellbaum, C. (dir.) (1998). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. Cambridge, MA: MIT Press.

Femmer, H., Méndez Fernández, D., Wagner, S. & Eder, S. (2017). Rapid quality assurance with Requirements Smells. *Journal of Systems and Software*, 123, 190 – 213.

Fernández, D. M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.-T., Greer, D., Lassenius, C., Männistö, T., Nayabi, M., Oivo, M., Penzenstadler, B., Pfahl, D., Prikladnicki, R., Ruhe, G., Schekelmann, A., Sen, S., Spinola, R., Tuzcu, A., de la Vara, J. L. & Wieringa, R. (2017, 5). Naming the pain in requirements engineering. *Empirical Software Engineering*, 2298–2338.

Ferrari, A., Dellorletta, F., Esuli, A., Gervasi, V. & Gnesi, S. (2017). Natural language requirements processing: A 4D vision. *IEEE Software*, 34(6), 28 – 35.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A. & Ré, C. (2023). Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In *The Eleventh International Conference on Learning Representations, ICLR*, Kigali, Rwanda,. OpenReview.net.

Gaaevic, D., Djuric, D., Devedzic, V. & Selic, B. (2006). *Model Driven Architecture and Ontology Development*. Berlin, Heidelberg: Springer-Verlag.

Garay-Vitoria, N. & Abascal, J. (2006). Text prediction systems: a survey. *Univers. Access Inf. Soc.*, 4(3), 188–203.

Garousi, V., Felderer, M. & Mäntylä, M. V. (2019). The role of grey literature in software engineering research. *Information and Software Technology*, 106, 101–121.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol, USA: O'Reilly Media, Inc.

Ghayoomi, M. & Momtazi, S. (2009). An overview on the existing language models for prediction systems as writing assistant tools. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, 5083–5087.

Gilardi, F., Alizadeh, M. & Kubli, M. (2023, 30). ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*.

Gnanasekaran, R. K., Chakraborty, S., Dehlinger, J. & Deng, L. (2021). *Using recurrent neural networks for classification of natural language-based non-functional requirements*.

Görnitz, N., Youssef, A. & Höppner, F. (2015). What is the State of the Art in Word Embeddings? In *Proceedings of the Workshop Events and Stories in the News*.

Guarino, N., Oberle, D. & Staab, S. (2009). *What Is an Ontology?* International Handbooks on Information Systems. Switzerland: Springer.

Hagiwara, M. (2020). *Real-World Natural Language Processing*. USA: Packt Publishing.

Hamarashid, H. K., Saeed, S. A. M. & Rashid, T. A. (2022). A comprehensive review and evaluation on text predictive and entertainment systems. *Soft Comput.*, 26(4), 1541–1562.

Heck, P. & Zaidman, A. (2014). A Quality Framework for Agile Requirements: A Practitioner's Perspective. *ArXiv, abs/1406.4692*.

Heng, S., Tsilionis, K. & Wautelet, Y. (2023). Building User Stories and Behavior Driven

Development Scenarios with a Strict Set of Concepts: Ontology, Benefits and Primary Validation. p. 1422 – 1429., Republic of Korea. ACM.

Henriksson, A. & Zdravkovic, J. (2020). *A Data-Driven Framework for Automated Requirements Elicitation from Heterogeneous Digital Sources*, Vol. 400 of *Lecture Notes in Business Information Processing*.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M. & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 51, 915–929.

Institute, P. M. (2017). *Agile practice guide*. Project Management Institute.

ISO (2018). ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, 1–104.

Jacobson, I. (1992). *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.

Jain, C., Anish, P. R., Singh, A. & Ghaisas, S. (2023). *A Transformer-based Approach for Abstractive Summarization of Requirements from Obligations in Software Engineering Contracts* [Conference paper].

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 11–21.

Jurafsky, D. & Martin, J. H. (2009). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*, 2nd Edition. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International.

Kaiya, H. & Saeki, M. (2006). Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In *14th IEEE International Conference on Requirements Engineering (RE 2006)*, 11-15 September 2006, Minneapolis/St.Paul, Minnesota, USA, 186–195. IEEE Computer Society.

Kang, Y., Li, H., Lu, C. & Pu, B. (2019). A transfer learning algorithm for automatic requirement model generation. *Journal of Intelligent and Fuzzy Systems*, 36(2), 1183 – 1191.

Kannan, V., Basit, M. A., Bajaj, P., Carrington, A. R., Donahue, I. B., Flahaven, E. L., Medford, R., Melaku, T., Moran, B. A., Saldana, L. E., Willett, D. L., Youngblood, J. E. & Toomay, S. M. (2019). User stories as lightweight requirements for agile clinical decision support development. *J. Am. Medical Informatics Assoc.*, 26(11), 1344–1354.

Kaur, K. & Kaur, P. (2024). SABDM: A self-attention based bidirectional-RNN deep model for requirements classification. *Journal of Software: Evolution and Process*, 36(2).

Khan, H. U., Niazi, M., El-Attar, M., Ikram, N., Khan, S. U. & Gill, A. Q. (2021). Empirical Investigation of Critical Requirements Engineering Practices for Global Software Development. *IEEE Access*, 9, 93593 – 93613.

Khan, J. (2024). A systematic mapping to investigate the application of machine learning techniques in requirement engineering activities. *CAAI Transactions on Intelligence Technology*.

Kitchenham, B., Budgen, D. & Brereton, P. (2015). *Evidence-Based Software Engineering and Systematic Reviews*. Innovations in Software Engineering and Software Development Series. Boca Raton, USA: Chapman & Hall/CRC.

Knapp, J., Zeratsky, J. & Kowitz, B. (2016). *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. Simon & Schuster.

Kokol, P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. *Information Switzerland*, 15(6).

Kumar, B., Tiwari, U. & Dobhal, D. C. (2022). User Story Splitting in Agile Software Development using Machine Learning Approach. In *PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing*, 167–171., Wagnaghat, India. IEEE.

Kustiawan, Y. A. & Lim, T. Y. (2023). User Stories in Requirements Elicitation: A

Systematic Literature Review. In *2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS)*, 211–216., Penang, Malaysia. IEEE.

Lam, L. K., Hurtado, C. A. L. & Portillo, L. W. (2022). Framework for automating requirement elicitation using a chatbot. In *Proceedings of the 2022 IEEE Engineering International Research Conference (EIRCON)*, 1–4., Peru. IEEE.

Lane, H., Hapke, H. & Howard, C. (2019). *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Manning Publications.

Laskar, M. T. R., Alqahtani, S., Bari, M. S., Rahman, M., Khan, M. A. M., Khan, H., Jahan, I., Bhuiyan, A., Tan, C. W., Parvez, M. R., Hoque, E., Joty, S. & Huang, J. (2024). *A Systematic Survey and Critical Review on Evaluating Large Language Models: Challenges, Limitations, and Recommendations*.

Lastra-Díaz, J. J., Goikoetxea, J., Hadj Taieb, M. A., García-Serrano, A., Ben Aouicha, M. & Agirre, E. (2019). A reproducible survey on word embeddings and ontology-based methods for word similarity: Linear combinations outperform the state of the art. *Engineering Applications of Artificial Intelligence*, 85, 645 – 665.

Li, Y., Shibata, H. & Takama, Y. (2019). Chatbot-mediated Personal Daily Context Modeling upon User Story Graph. In *2019 International Conference on Technologies and Applications of Artificial Intelligence, TAAI 2019, Kaohsiung, Taiwan, November 21-23, 2019*, 1–6. IEEE.

Lin, C.-Y. (2004, july). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81., Barcelona, Spain. Association for Computational Linguistics.

Liu, K., Reddivari, S. & Reddivari, K. (2022). Artificial Intelligence in Software Requirements Engineering: State-of-the-Art. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)*, 106–111.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M. & Brinkkemper, S. (2015). Forging high-quality User Stories: Towards a discipline for Agile Requirements. In D. Zowghi, V. Gervasi, & D. Amyot (dir.). *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, 126–135. IEEE Computer Society.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M. & Brinkkemper, S. (2016). Improving agile requirements: the Quality User Story framework and tool. *Requir. Eng.*, 21(3), 383–403.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M. & Brinkkemper, S. (2016). *The Use and Effectiveness of User Stories in Practice*, Vol. 9619 of *Lecture Notes in Computer Science*. Springer.

Madala, K., Gaither, D., Nielsen, R. & Do, H. (2017). Automated Identification of Component State Transition Model Elements from Requirements. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 386–392.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J. & McClosky, D. (2014). *The stanford CoreNLP natural language processing toolkit*.

Marques, N., Silva, R. & Bernardino, J. (2024). Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet*, p. 180.

Mateus, D., da Silveira, D. S. & Araújo, J. (2023). A Systematic Approach to Derive User Stories and Gherkin Scenarios from BPMN Models. *Lecture Notes in Business Information Processing*, 483 LNBIP, 235 – 244.

McElroy, K. (2017). *Prototyping for Designers: Developing the Best Digital and Physical Products*. O'Reilly Media, Incorporated.

Melis, G., Kočíský, T. & Blunsom, P. (2020). Mogrifier LSTM.

Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I. & Roth, D. (2024). Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey. *ACM Comput. Surv.*, 56(2), 30:1–30:40.

Mourão, E., Pimentel, J. F., Murta, L., Kalinowski, M., Mendes, E. & Wohlin, C. (2020). On the performance of hybrid search strategies for systematic literature reviews in software engineering. *Information and Software Technology*, 123, 106294.

Murtazina, M. & Avdeenko, T. V. (2019). An Ontology-Based Approach to the Agile Requirements Engineering. In N. S. Bjørner, I. B. Virbitskaite, & A. Voronkov (dir.). *Perspectives of System Informatics - 12th International Andrei P. Ershov Informatics Conference, PSI 2019, Novosibirsk, Russia, July 2-5, 2019, Revised Selected Papers*, Vol. 11964 of *Lecture Notes in Computer Science*, 205–213. Springer.

Nistala, P. V., Rajbhoj, A., Kulkarni, V., Soni, S., Nori, K. V. & Reddy, R. (2022). Towards digitalization of requirements: generating context-sensitive user stories from diverse specifications. *Autom. Softw. Eng.*, 29(1), 26.

OpenAI (2024). *Fine-tuning Guide: When to Use Fine-Tuning*. Accessed: 2025-04-28, Retrieved from <https://platform.openai.com/docs/guides/fine-tuning>

Ordoñez, H., Villada, A. F. E., Vanegas, D. L. V., Lozada, C. A. C., Ordóñez, A. & Segovia, R. (2015). An Impact Study of Business Process Models for Requirements Elicitation in XP. *9155*, 298–312.

Panichella, S. & Ruiz, M. (2020). Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback. In *28th IEEE International Requirements Engineering Conference, RE 2020*, 404–407., Zurich, Switzerland. IEEE.

Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J. (2002, july). Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318., Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Peña Veitía, F. J., Roldán, L. & Vegetti, M. (2020). User Stories identification in software's issues records using natural language processing. In *2020 IEEE Congreso Bienal de Argentina, ARGENCON 2020 - 2020 IEEE Biennial Congress of Argentina, ARGENCON 2020*.

Pham, P., Nguyen, L. T. T., Pedrycz, W. & Vo, B. (2022). Deep learning, graph-based

text representation and classification: a survey, perspectives and challenges. *Artificial Intelligence Review*, 56(6).

Pham, T. B. (2018, Nov). *Proof of concept in software development: 5 very important factors*.

Pokharel, P. & Vaidya, P. (2020). A Study of User Story in Practice. In *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*, 1–5.

Popay, J., Roberts, H., Sowden, A., Petticrew, M., Arai, L., Rodgers, M., Britten, N., Roen, K. & Duffy, S. (2006). *Guidance on the conduct of narrative synthesis in systematic reviews*. A product from the ESRC Methods Programme.

Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018). Improving language understanding by generative pre-training. *1*.

Raharjana, I. K., Siahaan, D. O. & Fatichah, C. (2019). User Story Extraction from Online News for Software Requirements Elicitation: A Conceptual Model. In *16th International Joint Conference on Computer Science and Software Engineering, JCSSE 2019, Chonburi, Thailand, July 10-12, 2019*, 342–347. IEEE.

Raharjana, I. K., Siahaan, D. O. & Fatichah, C. (2021). User Stories and Natural Language Processing: A Systematic Literature Review. *IEEE Access*, 9, 53811–53826.

Rajender Kumar Surana, C. S., Shriya, Gupta, D. B. & Shankar, S. P. (2019). Intelligent Chatbot for Requirements Elicitation and Classification. In *2019 4th International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT)*, 866–870.

Rakib, O. F., Akter, S., Khan, M. A., Das, A. K. & Habibullah, K. M. (2019). Bangla Word Prediction and Sentence Completion Using GRU: An Extended Version of RNN on N-gram Language Model. In *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 1–6.

Ramesh, B., Cao, L. & Baskerville, R. L. (2010). Agile requirements engineering practices

and challenges: an empirical study. *Inf. Syst. J.*, 20(5), 449–480.

Raschka, S. (2024). *Build a Large Language Model (From Scratch)*. From Scratch. Shelter Island, NY, USA: Manning.

Resketi, M. R., Motameni, H., Nematzadeh, H. & Akbari, E. (2020). Automatic summarising of user stories in order to be reused in future similar projects. *IET Softw.*, 14(6), 711–723.

Rodeghero, P., Jiang, S., Armaly, A. & McMillan, C. (2017). Detecting user story information in developer-client conversations to generate extractive summaries. In S. Uchitel, A. Orso, & M. P. Robillard (dir.). *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*, 49–59. IEEE / ACM.

Ronanki, K., Berger, C. & Horkoff, J. (2023). Investigating ChatGPT’s Potential to Assist in Requirements Elicitation Processes. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 354–361.

Ronanki, K., Daniel, B. C. & Berger, C. (2023). ChatGPT as a Tool for User Story Quality Evaluation: Trustworthy Out of the Box? *489*, 173–181.

Santos, N., Pereira, J., Morais, F., Barros, J., Ferreira, N. & Machado, R. J. (2018). Deriving user stories for distributed scrum teams from iterative refinement of architectural models. In A. Aguiar (dir.). *Proceedings of the 19th International Conference on Agile Software Development, XP 2019, Companion, Porto, Portugal, May 21-25, 2018*, 40:1–40:4. ACM.

Schneider, F. & Berenbach, B. (2013). A Literature Survey on International Standards for System Requirements Engineering. *Procedia Computer Science*, 796–805.

Schröder, M. (2023). AutoScrum: Automating Project Planning Using Large Language Models.

Schwaber, K. & Sutherland, J. (2020). *The 2020 Scrum Guide™*. Retrieved 2023-04-30, from <https://scrumguides.org/scrum-guide.html>

Schön, E.-M., Thomaschewski, J. & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards and Interfaces*, 49, 79–91.

Scoggin, S. B. & Torres Marques-Neto, H. (2024). Identifying Valid User Stories Using BERT Pre-trained Natural Language Models. In A. Rocha, H. Adeli, G. Dzemyda, F. Moreira, & V. Colla (dir.). *Information Systems and Technologies*, 167–177., Cham. Springer Nature Switzerland.

Shini, R. & Kumar, V. (2021). Recurrent Neural Network based Text Summarization Techniques by Word Sequence Generation. *Proceedings of the 6th International Conference on Inventive Computation Technologies, ICICT 2021*, 1224–1229.

Shore, J. & Warden, S. (2021). *The Art of Agile Development*. O'Reilly Media.

Siahaan, D., Raharjana, I. K. & Fatichah, C. (2023). User story extraction from natural language for requirements elicitation: Identify software-related information from online news. *Information and Software Technology*, 158.

So, D. R., Manke, W., Liu, H., Dai, Z., Shazeer, N. & Le, Q. V. (2021). Primer: Searching for Efficient Transformers for Language Modeling. *CoRR*, [abs/2109.08668](https://arxiv.org/abs/2109.08668).

Soam, M. & Thakur, S. (2022). Next Word Prediction Using Deep Learning: A Comparative Study. In *2022 12th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 653–658.

Soeken, M., Wille, R. & Drechsler, R. (2012). Assisted Behavior Driven Development Using Natural Language Processing. In C. A. Furia & S. Nanz (dir.). *Objects, Models, Components, Patterns*, 269–287., Berlin, Heidelberg. Springer Berlin Heidelberg.

Sommerville, I. (2011). *Software Engineering*. International Computer Science Series. Pearson.

Sommerville, I. (2016). *Software Engineering* (10th). Always learning. Boston, MA: Pearson.

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M. Z., McDermid, J. A. & Paige, R. F. (2012, 7). Large-scale complex IT systems. *Commun. ACM*, 71–77.

Sparck Jones, K. (1972, 1). A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. *Journal of Documentation*, 11–21.

Sridhara, G., G., R. H. & Mazumdar, S. (2023). ChatGPT: A Study on its Utility for Ubiquitous Software Engineering Tasks.

Tan, Z., Wang, M., Xie, J., Chen, Y. & Shi, X. (2018). Deep Semantic Role Labeling With Self-Attention. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, 4929–4936., New Orleans, Louisiana, USA. AAAI Press.

Tay, Y., Dehghani, M., Bahri, D. & Metzler, D. (2022, 6). Efficient Transformers: A Survey.

Taylor, S. (2020). *Markov Models: An Introduction to Markov Models*. Steven Taylor.

Thamrongchote, C. & Vatanawood, W. (2016). Business process ontology for defining user story. In *15th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2016, Okayama, Japan, June 26-29, 2016*, 1–4. IEEE Computer Society.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *ArXiv, abs/2302.13971*.

Umar, M. A. & Lano, K. (2024). Advances in automated support for requirements engineering: a systematic literature review. *Requirements Engineering*, 29(2), 177 – 207.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2023). *Attention Is All You Need*.

Wagner, S., Fernández, D. M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl,

D., Conte, T., Christiansson, M.-T., Greer, D., Lassenius, C., Männistö, T., Nayebi, M., Oivo, M., Penzenstadler, B., Prikładnicki, R., Ruhe, G., Schekelmann, A., Sen, S., Spínola, R., Tuzcu, A., Vara, J. L. D. L. & Winkler, D. (2019, 2). Status Quo in Requirements Engineering: A Theory and a Global Family of Surveys. *ACM Trans. Softw. Eng. Methodol.*

Wake, B. (2003, Aug). *Invest in good stories, and Smart Tasks*. Retrieved from <https://xp123.com/invest-in-good-stories-and-smart-tasks/>

Wake, W. (2002). *Extreme Programming Explored*. XP series. Addison-Wesley.

Webber, E. (2023). *Pretrain Vision and Large Language Models in Python: End-To-end Techniques for Building and Deploying Foundation Models on AWS*. United Kingdom: Packt Publishing, Limited.

Wieggers, K. & Beatty, J. (2013). *Software Requirements*. Pearson Education.

Wieringa, R., Maiden, N., Mead, N. & Rolland, C. (2006). Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requirements Engineering*, 11(1), 102 – 107.

Witten, I. H., Frank, E. & Hall, M. A. (2011). *Data mining: practical machine learning tools and techniques, 3rd Edition*. USA: Morgan Kaufmann, Elsevier.

Wohlin, C. (2014). A Snowballing Procedure for Systematic Literature Studies and a Replication. In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 321–330.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. & Wesslén, A. (2012). *Experimentation in software engineering*. Berlin, Heidelberg: Springer Science & Business Media.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C. & Regnell, B. (2012). *Experimentation in Software Engineering*. Heidelberg, Berlin, Germany: Springer.

Zhang, H. & Babar, M. A. (2013). Systematic reviews in software engineering: An

empirical investigation. *Inf. Softw. Technol.*, 55(7), 1341–1354.

Zhang, J., Chen, Y., Niu, N., Wang, Y. & Liu, C. (2023). *Empirical Evaluation of ChatGPT on Requirements Information Retrieval Under Zero-Shot Setting*.

Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q. & Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E.-V. & Batista-Navarro, R. T. (2021). Natural Language Processing for Requirements Engineering. *ACM Computing Surveys*, 54(3), 1–41.

Zhou, X., Jin, Y., Zhang, H., Li, S. & Huang, X. (2016). A Map of Threats to Validity of Systematic Literature Reviews in Software Engineering. In A. Potanin, G. C. Murphy, S. Reeves, & J. Dietrich (dir.). *23rd Asia-Pacific Software Engineering Conference, APSEC 2016, Hamilton, New Zealand, December 6-9, 2016*, 153–160. IEEE Computer Society.

APPENDIX A
ETHICS CERTIFICATION

This study was conducted in compliance with ethical standards and approved by *Le Comité d'éthique de la recherche de l'Université du Québec à Chicoutimi (CER-UQAC)*. The ethical certificate number is 2023-1326.