2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA) | 978-1-6654-8350-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/SERA54885.2022.9806783

Automatic Transformation of Natural to Unified Modeling Language: A Systematic Review

Sharif Ahmed
Dept. of Computer Science
Boise State University
Boise, USA
sharifahmed@u.boisestate.edu

Arif Ahmed
Dept. of Computer Science
Boise State University
Boise, USA
arifahmed@u.boisestate.edu

Nasir U. Eisty
Dept. of Computer Science
Boise State University
Boise, USA
nasireisty@boisestate.edu

***Abstract*—*Context:* Processing Software Requirement Specifications (SRS) manually takes a much longer time for requirement analysts in software engineering. Researchers have been working on making an automatic approach to ease this task. Most of the existing approaches require some intervention from an analyst or are challenging to use. Some automatic and semi-automatic approaches were developed based on heuristic rules or machine learning algorithms. However, there are various constraints to the existing approaches to UML generation, such as restrictions on ambiguity, length or structure, anaphora, incompleteness, atomicity of input text, requirements of domain ontology, etc. . *Objective:* This study aims to better understand the effectiveness of existing systems and provide a conceptual framework with further improvement guidelines. *Method:* We performed a systematic literature review (SLR). We conducted our study selection into two phases and selected 70 papers. We conducted quantitative and qualitative analyses by manually extracting information, cross-checking, and validating our findings. *Result:* We described the existing approaches and revealed the issues observed in these works. We identified and clustered both the limitations and benefits of selected articles. *Conclusion:* This research upholds the necessity of a common dataset and evaluation framework to extend the research consistently. It also describes the significance of natural language processing obstacles researchers face. In addition, it creates a path forward for future research.**

***Index Terms*—Requirement Elicitation, Software Engineering, Natural Language Processing, Unified Modeling Language**

I. Introduction

Requirements analysis in software development is an essential and rudimentary task. The quality analysis of requirements elicitation and further developing software work-products such as various diagrams and formatted textual descriptions leads to a successful project, product, or service. We human beings, as an analyst, have the innate power to understand the information in texts contextually regardless of the existence of misspelled words, incorrect grammar, and indirectly stated items. This performance has not been obtained yet in Natural Language Processing (NLP) or Artificial intelligence, and it is a long way to go. But we human beings are also limited by fatigue, drowsiness, distractions, fluctuation in concentrations, which makes us error-prone despite our aforementioned power. So, the machines' ability to work relentlessly with error and humans' ability to work perfectly for short periods is not as easy to put on two pans of a weighing scale's beam and come to a conclusion. Berry et al. [1] suggested that NLP tools for requirements engineering can be an effective approach to activities that are tedious and rigorous.

Unified Modeling Language (UML) is "a general-purpose visual modeling language to specify, visualize, construct, and document the artifacts of a software system"- Rumbaugh et al. [2]. There is no fully working method or tool to generate UML diagrams from the informal natural language (NL) requirements. Most existing approaches have high-level complexity, are computationally costly, or have limitations. Researchers found that earlier approaches require developer intervention for UML diagram creation. Besides, there are a few approaches that are comparatively recent and are fully automated. Still, these approaches are working under some constrained input such as restricted NL or some particular form of texts [3]. A previous work takes informal NL text requirements and generates UML Use Case and Activity Diagrams without any developer's intervention or assistance [4]. Any fully automatic approach or tool can significantly help with requirement elicitation and modeling for software analysts. It can also save the overall cost and time within a software development process.

Many tools and approaches have been proposed and developed to extract information from natural language text to generate UML diagrams. However, natural language processing has many problems such as ambiguity, uncertainty, incompleteness, and incoherence [3]. Existing approaches perform poorly at times because of this ambiguity of NL. On top of that, large requirement documents can produce other issues while processing. Besides, crucial information is sometimes missed, which feeds the verb to develop complete understanding. A software analyst has to figure out such problems and fix these issues using her domain knowledge. Many tools and techniques have been proposed to solve this kind of problem. But in reality, these tools are not being used in Software Development Life-Cycle for many constraints. And most of the tools require frequent interventions of an analyst to finish the process. In addition, the majority of tools are limited to producing class diagrams only [5], [6].

Though there are some new solutions, they still require more rules to increase the domain knowledge, which is another problem [3], [7]. So, our work will provide a conceptual framework showing an overview of using rule-based heuris-

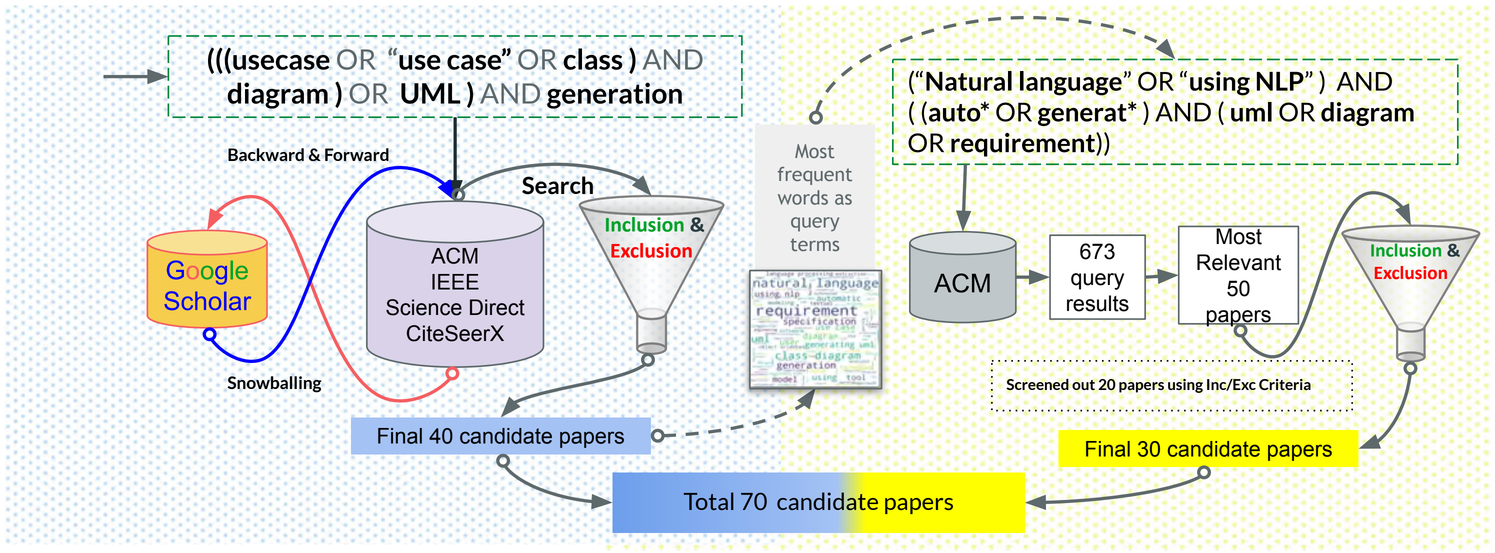

(((usecase OR "use case" OR class) AND diagram) OR UML) AND generation
Backward & Forward
Search
Google Scholar
ACM IEEE Science Direct CiteSeerX
Inclusion & Exclusion
Snowballing
Final 40 candidate papers
Most frequent words as query terms
("Natural language" OR "using NLP") AND ((auto* OR generat*) AND (uml OR diagram OR requirement))
ACM
673 query results
Most Relevant 50 papers
Inclusion & Exclusion
Screened out 20 papers using Inc/Exc Criteria
Final 30 candidate papers
Total 70 candidate papers

Fig. 1. Study selection process

tic approaches and machine learning-based approaches. This framework will help researchers to make decisions while finding a new solution.

II. Related Work

Though there are not many systematic studies related to our research questions, we found a few notable literature reviews that we discuss in this section.

Zaho et al. [8] conducted a robust systematic literature review on NLP for requirements engineering (RE), capturing almost every aspect of RE tasks: detection, extraction, classification, modeling, tracing and relating, search, and retrieval. They covered the breadth of RE, which is valuable for the software research community. In contrast, we give a depth of RE regarding NL to UML automation. As they covered on breadth, it is not expected to have a depth of any of the tasks mentioned above they covered.

Omer et al. [9] performed a survey that addressed techniques and outputs. Their analysis of strengths and weaknesses is limited to the number of diagrams each study generated.

Esra et al. [3] conducted another systematic mapping study on the techniques and approaches of NL to UML Class diagram. However, our analysis is neither limited to class diagrams nor mapping the techniques and approaches. Instead, we dug deeper into the existing approaches, tools, pros, and cons. In addition, we cover what the primary studies addressed, attempted, or overlooked and their success and failure anecdotes from their published works.

III. Research Methodology

In this section, we discuss our research methodology for this study. At first, we developed a research protocol following guidelines for systematic literature reviews by Kitchenham [10] to make our research process uncompromising and evident. Then, we followed the following steps: research question formation, study selection, data extraction, and data synthesis. Table I shows the protocol overview that we developed and executed.

A. *Research Questions*

Our research motivation and objectives led us to the following research questions:

RQ1: What are the existing approaches to automate the UML generation? The answer to this question will recapitulate the existing tools and techniques used by the software engineering community.

RQ2: How effective are the existing approaches? This inquiry will portray opportunities and obstacles of ongoing effort to solve NL to UML transformations.

B. *Searching Strategy*

We mainly focused following digital libraries for our study:

- ACM Digital Library (ACM)
- IEEE Xplore Digital Library(Xplore)
- Science Direct Digital Library (Elsevier)
- CiteSeerX
- Google Scholar

Our study comprises two phases. At first, we searched on aforementioned digital libraries using the query string, *(((usecase OR "use case" OR class) AND diagram) OR UML) AND generation*, addressing the most commonly used UML diagrams. We snowballed backward and forward with the retrieved candidate papers on Google Scholar. At that point, we selected 40 articles perusing the full articles. Then we checked a shallow synthesis of words from the titles and the abstracts of these papers and identified the top frequent words. In the second phase, using these words, we built another search string *("Natural language" OR "using NLP") AND ((auto* OR generat*) AND (uml OR diagram OR requirement))* and performed the second phase of our search on abstracts only. We chose only ACM Digital Library in this phase as it retrieved 673 articles, the least number of articles than other libraries. We manually selected 30 articles after screening the title and abstract from these retrieved articles. Our searching period was Sep 2021 to Dec 2021, and the publishing years of papers were from 1994 to 2021.

TABLE I
PROTOCOL SUMMARY

Research Questions	RQ1: What are the existing approaches to automate the UML generation? RQ2: How effective are the existing approaches?
Search string	**Phase-1:** (((usecase OR "use case" OR class) AND diagram) OR UML) AND generation **Phase-2:** ("Natural language" OR "using NLP") AND ((auto* OR generat*) AND (uml OR diagram OR requirement))
Search strategy	**Phase-1:** DB search: ACM, Google Scholar, IEEE, ScienceDirect, Springer, CiteSeerX Backward and forward snowballing using Google Scholar **Phase-2:** DB search: ACM Formulated new query string from most frequent words found in the titles of articles filtered in Phase-1
Inclusion criteria	The article is written in English The article is published in a scholarly journal or conference/ workshop/ symposium proceedings The article covers at least one of our RQs partially or fully
Exclusion criteria	The article doesn't answer any of our RQs either partially or fully The article is written in Non-English Any article retracted from publisher The article is secondary study derived from primary studies i.e. Systematic Literature Review or Mapping Study
Study type	Primary studies

TABLE II
DATA EXTRACTION FORM

Field	**Categories**	**Relvant RQ**
Paper Title	Free text	-
Year	Number	Demographics
Source	Venue / Journal / Conference	Demographics
Authors	Free text	Demographics
Abstract	Free text	-
Relationships	(Aggregation, Inheritance, Generalization, Association, Composition, Dependency, Multiplicity, Inclusion, Exclusion)	RQ2
Methodology	Analysis of their implementation	RQ2
Pros	Analysis of their resolved problem	RQ2
Cons	Analysis of their unresolved problem	RQ2
Automation	Automatic, Semi-Automatic, Manual	RQ1
Approach	Heuristic Rule based, Machine Learning based, Hybrid (ML+HR)	RQ1
Output	Type of output(UML)produced	RQ1 , RQ2
Technology	Analysis of the tools, technology used to solve the problem	RQ2
Evaluation	Free text	RQ1, RQ2

C. Quality Assessment and Inclusion/Exclusion

We considered the studies written in English; published in a scholarly SE journal, conference, workshop, or symposium proceedings. We checked if any of the studies already answered any of our RQs partially or fully for assessing quality. We excluded secondary studies, i.e., Systematic Literature Review and Systematic Mapping Study. We also deduplicated the papers as depicted in Fig. 1.

D. Data Extraction

As our final primary studies were selected, we developed a data extraction form to answer our RQs. Table II shows data fields and their mapping to RQs. The first two authors manually went through each of the papers and extracted information from each of the primary studies to fillup the form. The third author reviewed the results. Then we discussed the emerged conflicts and fixed them.

E. Data Synthesis

The data extraction phase led us to synthesize quantitative and qualitative results. Having extracted the data in the form, we synthesized it to discover:

- Year-wise distribution of published studies
- Statistics of approaches to convert NL to UML
- Statistics of automation to convert NL to UML
- Relationships among UML components resolved
- Distribution of final/intermediate outputs acquired
- Usage of technologies or tools to solve the problems
- Analysis of dataset, metric, and evaluation techniques
- Analysis of strength and limitation of existing solutions

The Tables III, IV, V, and VI and Fig 2 provide quantitative insights. Section 4.2 provides 16 facets with both quantitative and qualitative insights.

IV. RESULTS

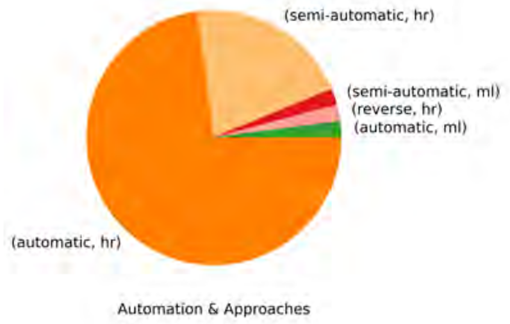
We discuss our findings according to the research questions in this section.

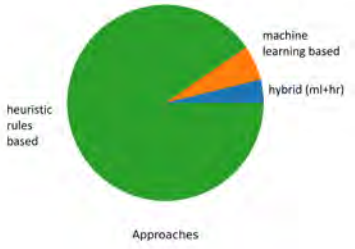
A. RQ1: What are the existing approaches to automate the UML generation?

At first, we grouped all the approaches found in our primary selected papers into the following categories: Heuristic Rule-Based, Machine Learning, Automatic, Semi-automatic, and Hybrid. Fig. 2 shows the approaches and automation we found in our analysis. But as we moved forward, we figured out that many research works combined these approaches. So, we refined our cluster represented in Fig. 2. We found that most of the work was carried out by applying the heuristic rules, and a few other works used machine learning techniques. In addition, some of the papers obtained significant output by using both heuristic rules and machine learning techniques.

1) UMLs Generated: Table III shows the UML diagrams that existing studies covered. Table IIIpresents the most common UML outputs such as usecase, sequence, and activity. Among them, the class diagram is studied the most.

2) Technologies Used: We have manually extracted the technologies used from each of the papers from our primary studies. Table IV shows the list of technologies used. Researchers used a variety of NLP techniques in their studies. We observed significant use of the Stanford CoreNLP library. Besides, we also found the use of OpenNLP and SharpNLP. Stanford's POS tagger, Brill, TreeTagger, dependency parser,





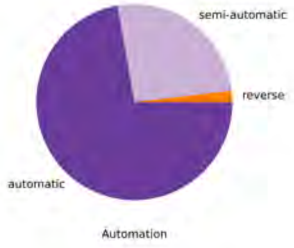


Fig. 2. Approaches to solve NL to UML transformation

Here, **hr**: Heuristic rules based approach, **ml**: Machine learning approach

TABLE III
OUTPUTS OBTAINED IN STUDIES

Outputs	**Studies**
class diagram	[4]–[6], [11]–[39]
usecase diagram	[7], [13], [40]–[45]
object diagram	[30], [46]–[49]
processed sentences	[50]–[53]
sequence diagram	[11], [14], [20], [45]
comparison	[8], [54]–[56]
code	[11], [17], [35]
use case	[37], [57]
test case	[35], [57]
processed srs	[58], [59]
activity diagram	[7], [60]
meta model	[57], [61]
Other Outputs: traceability [44]; graph [62]; sbvr [63]; er-diagram [64]; processed named entity [65]; b-spec [66]; owl class [67]; proposal [68]; collaboration diagram [13]; test cases [57]; natural language [69]; feature diagram [70]; gui prototype [71]	

TABLE IV
TECHNOLOGY USED IN EXISTING STUDIES

Technology	**Studies**
nlp	[5], [8], [11], [12], [14], [15], [17]–[19], [25]–[31], [33], [36], [40], [42]–[47], [49], [51]–[53], [56]–[58], [60], [62], [63], [65], [68], [71]
rule	[1], [15], [17], [21], [23], [24], [29]–[31], [33], [44], [49]
pos tagger	[1], [5], [6], [15], [19], [25]–[28], [40]–[42], [53], [64]
parse	[1], [13], [21], [28], [41], [42], [45], [46], [54], [64], [68]
stanford corenlp	[1], [7], [11], [13]–[15], [17], [43], [45], [54], [62], [68]
ontology	[1], [5], [12], [18], [27]–[29], [42], [65]
parser	[13], [21], [28], [42], [45], [46], [54], [68]
wordnet	[5], [11]–[13], [27], [28], [54], [59]
tree	[34], [41], [42], [54], [64], [69]
gui	[5], [18], [25], [47], [52], [59]
open nlp	[12], [25], [27], [28], [68]
dependency	[16], [22], [34], [45]
sbvr	[15], [15], [24], [24]
graph	[22], [34], [62]
ml	[20], [36], [70]
ocl	[20], [34], [57]
traceability	[25], [26]
Other Technologies: brill [5]; treetagger [42]; bayes [40]; featureide [70]; gkps [16]; javarap [13]; reasoning [19]; sharpnlp [52]; spacy [53]; spider [5]; verbnet [59]; semnet [38]; lolita [38]; rtool [37];	

POS parser, and Stanford's parser were most commonly used for Parts of Speech (POS) tagging. WordNet was most common for resolving ambiguity, but WSD and VerbNet were also used. A set of heuristic rules were defined for heuristic-rule-based solutions. We also found several works using the domain ontology technique.

3) Preprocessing techniques applied: We found that all the preprocessing treatments for NL are almost similar in nature. The first task in preprocessing is to split the text into sentences. Then the sentences are tokenized using either stemming or lemmatization. Surprisingly we found several works that applied both techniques together [3], [4], [7]. In stemming, the output can be meaningless words, affecting the use of tagger and parser. Then POS tagger and POS parser are used for identifying NP, VB, grammatical structures, etc. Then a set of rules is applied to identify actors, use cases, relationships, etc. But while normalizing the NL text automatically, there is a high chance of losing information [13]. So, the decision and treatment selection in preprocessing phase affects the overall performance.

B. *RQ2: How effective are the existing approaches?*

In this study, one of our objectives was to find "to what extent problem(s) were solved" and "what limitations were observed" from our primary studies. To do so, we extracted both pros and cons from each study. Then, we framed our findings into a conceptual framework comprising sixteen different facets in Fig. 3. Finally, we describe how far researchers could solve existing problems and what problems they could not solve while translating NL to UML in this section.

1) Ambiguity: The requirement document contains text where comprehension or meaning extracted by readers may vary based on readers' perspectives. In particular, requirement analysts can resolve the ambiguity from text with their domain knowledge. But when it comes to NLP, it becomes much harder to capture the correct meaning of a word by resolving ambiguity. Any misinterpretation of a word may produce defects that can affect the overall context [72]. Some studies solved ambiguities but mostly resolved lexical ambiguity. In NLP, syntactic or semantic ambiguities are also common problems. But, existing approaches could not resolve such ambiguities completely [37], [38], [49], [55], [56], [58], [59].

TABLE V
RELATIONSHIPS RESOLVED IN EXISTING STUDIES

Relation Resolution	Studies
aggregation	[3]–[6], [12], [14]–[16], [24], [28], [29], [31], [37], [42], [43], [46], [64], [72], [75]
association	[4], [6], [11], [12], [17], [23], [24], [29]–[31], [37], [38], [43], [46], [65], [70], [72], [75], [76]
generalization	[3]–[6], [12], [15], [23], [24], [28], [29], [31], [37]
composition	[4], [12], [24], [28], [42], [43], [64], [76]
multiplicity	[6], [24], [29], [30], [42]
dependency	[3], [4], [24], [28]
inheritance	[24], [43]

A study stated six kinds of ambiguity in software requirements: lexical, syntactic, semantic, pragmatic, vagueness, and language error [73]. However, in their work, they only focused on lexical, syntactic, and language errors only [59]. We found some manual glossaries, machine learning, and ontology-based approaches to reduce ambiguity from the SRS. In addition, we found some tools for identifying and eliminating ambiguities either fully or partially. Tools are- WSD, QuaARS, ARM, RESI, SREE, NAI, SR-Elicitor, and NL2OCL [74].

2) Semantic Correctness: We found several studies resolved semantic incorrectness by using WordNet. But, in a study, they considered frequency counts while using WordNet caused misclassification [12], [28], [44].

3) Language: We did not find any study processing Non-English requirement text. We only found a work where the requirement text was translated from French into English [54].

4) Heuristic Rule: Studies using heuristic rules have shown promising work progress. However, most of them addressed the necessity of more heuristic rules for better performance [4], [7].

5) Relation Resolution: Several relationships connect the UML components. For instance, the association relationship connects each actor and the use case. Meryem et al. [42] identified the actors first and then discovered the use cases using the linkage from the actors. And for more than one property associated with a concept or candidate, it is considered as a class; otherwise, an attribute [5]. Prepositions are emphasized for identifying relationships and associations [17]. Verb phrases are used to determine inheritance, association, composition, and aggregation relations to extract actors or classes; association and composition relations are used for identifying actions, operations, or methods [43]. Another type of relationship is semantic relation comprising vertical and horizontal relationships. Vertical relations include broader, part-of, or instance-of. Horizontal relations are similarity or relatedness [5]. SBVR, which has Object Oriented information, easily retrieves the association, multiplicity, aggregations, generalizations, and instances. A few others used Named-Entity-Recognition, Stanford Open Information Extraction, Ontology, WordNet, and Dependency Parser to solve these relations.

6) Text Restriction: We found a few papers that mentioned different constraints on NL text [40], [52]. We also investigated the structure of the NL text used for transforming into UML. Some studies put restrictions on:

- Text length: No study explicitly mentioned the text length except Sandeep et al. [40]. On average there were 10 sentences per story which are comparatively shorter. In reality, the original user stories are much more lengthy and lack clarity.
- Sentence length: Some approaches only used 5-15 sentences with an average sentence length of 10 words in their user requirement document. Some studies observed that long sentences having more than one N (Nouns) or VB (Verbs) can create disturbingly long use cases [40].
- Grammatical structure: The constraint to use modal verbs [52] or active sentences [13], [27], [44], [77] only in requirement text imposes additional restrictions.
- Controlled language: A few approaches were carried out using controlled NL instead of NL [15], [55], [63].

7) Formal Expression Extraction: In formal expression extraction, an expression must be consistent with the formal representation of the system. Dependency graph construction based on grammatical structure and abstract syntax trees can show similarities. But, synonyms applied within a sentence cause major problem [59]. For example, "CPU" and "Processor" can be used within the same sentence. In that case, Word Sense Disambiguation (WSD) is a good choice to resolve such issues.

8) Dataset: Most of the primary studies applied their models to one or more NL software requirement text(s) (aka case-study) to demonstrate the models' performance. We found input NL case-study appeared more than twice are: Library [12], [15], [16], [24], [31], [36], [64], ATM [37], [45], [60], [76], Elevator [45], [46], [60], Banking [22], [25], [26], [36], Home [53], [70], Arena [45], [60]. However, some studies didn't even address their input case-study [6], [14], [15], [40], [66], [76]. The number of input case-study used in the experiment ranges from 0 to 7.

Meryem et al. [42] worked with NLP techniques along with OWL ontology and MAT, then simulated the performance, including accuracy, recall, precision on classifying actor, use case, and relationships on a dataset that is obsolete and inaccessible. There are a few papers that mentioned the dataset they used. But we did not find all of them available publicly. We also found a few studies using significantly small and poor datasets [15], [35].

9) Anaphora Resolution: This is one of the most common and challenging problems in NLP. A few studies solved the only basic or pronominal issues using JavaRAP by replacing proper nouns with their correct noun form [13], [49]. For example: "John found the love of his life" where 'his' refers to 'John'.

10) Incompleteness: Requirement text with correct grammatical structure may have an information gap if the document is written considering readers understandability. Any human reader having field expertise can comprehend such incompleteness. But, it is difficult to capture such information gaps using incorporated NLP techniques [65], [72].

11) Atomicity: Another issue comes with a lack of atomicity which appears when the writer keeps the text unambiguous

TABLE VI
METRICS USED FOR EVALUATION IN EXISTING STUDIES

Metrics	Studies
precision	[6], [15], [16], [24], [25], [28], [29], [31], [36], [38], [40], [42], [53], [54], [63], [70], [76], [78], [79]
recall	[6], [15], [16], [24], [25], [28], [29], [31], [36], [38], [40], [53], [63], [70], [76], [78], [79]
enumeration	[22], [25], [32], [57], [76]
accuracy	[14], [40], [41], [45], [59]
relationship	[4], [6], [29], [70]
false positive	[36], [53], [79]
over specification	[6], [16], [31]
output type	[4], [7], [13]
false negative	[36], [53], [79]
completeness	[58], [60]
f-measure	[36], [63]
Other Metrics: weighted average [63], f-measure [63], ambiguity [58], [59], over-generalization [29], consistency [60], automation [13], atomicity [58], conceptual density of dataset [78], limitation [37], verifiability [59], computational time [58]	

by providing all the information to make the text complete but addresses more than one thing. In NLP, if we can split a sentence into smaller sentences, then it doesn't go along with atomicity. A few studies solved this problem by splitting a sentence into smaller sentences. But, for complex sentences, it required human intervention [72].

12) Misclassification: Researchers found a high level of misclassification for classes and attributes for using simple heuristic rules [16], [54]. Abdelkareem et al. [14] normalized the text before POS tagging then used Stanford NLP POS Tree. But, they found that the Stanford PoS tree misses Sub (Subject) by misclassifying VB (Verb)/NP (Noun Phrase).

13) Manual Intervention: We found studies producing outcomes with and without manual intervention. A few studies require it mandatorily, some of them require it to some extent [1], [7], [11], [30].

14) SRS Traceability: After identifying class, attributes, and operations (UML components), identified UML components are checked in a backward fashion to see whether the UML components are meaningful and working or not. This iterative process provides meaningful class diagrams [25]–[27].

15) Proof of Concepts: Many of the studies provided various software, IDE plugins, or other forms of deliverables, which are essential for the research community to believe their demonstrations and feasibility [12], [14], [17], [47], [63], [71].

16) Evaluation: The most common metrics found in our primary studies are based on counting proper classification or extraction of UML components from NL such as Precision, Recall Accuracy, F-measure, Enumeration, Over-specification, Over-generalization False Negative, False Positive. A few other less common metrics found are Completeness, Ambiguity, Conceptual Density of Dataset, Computational time, Weighted Average (in addition to enumeration), Verifiability, Consistency, and Atomicity. Some studies just contrasted their implementation with others as an evaluation such as Relationship, Output Type, Automation.

V. Discussion

We have found most papers using a rule-based heuristic approach which indicates the common trend for solving, but machine learning approaches are also noticeable in recent years. Each of the studies had taken an almost similar treatment for NL preprocessing using different libraries.

We observed many patterns in the existing research works and discovered trends and relationships. As described in section IV, we identified sixteen facets of the solutions and limitations of the existing approaches. We found the class diagram as the most used UML diagram. Implementing other UMLs like use case or usecase diagrams does require additional tasks and effort. We also found a comparatively low number of research mentioning their proof of concepts, but in some cases, these were neither publicly accessible nor found on the web. The primary concern for most of the studies was the ambiguity of NL. Few works have significantly mentioned this problem, its limitations, and its treatment.

However, most implementations had constraints such as satisfying a specific grammatical structure, using a domain ontology, sentence length, and absence of ambiguity or anaphora. Some also failed to identify UML components or detect relationships among them in some cases. Many evaluation metrics and datasets emerged in this study. Claims from evaluations are not comparable in many cases because of differences in metrics or datasets. Moreover, the case studies used lack more extensive or diverse descriptions.

VI. Threats to Validity

We did not use SpringerLink and Scopus due to not having access to these digital libraries within the period of this study. The SLR was independently executed by the first two authors and reviewed by the third author to minimize personal bias. We might have some selection bias in phase-1. We analyzed the most frequent words from the titles of selected 40 papers from phase-1. Using these most frequent words as query terms, we again searched the same digital libraries. Thus, the later searching phase minimizes query term selection bias as we constructed it from statistics instead of our choice. Among them, ACM digital library retrieved the least number of papers. So we considered only ACM for the second phase and manually selected the most relevant 30 papers for our second phase study selection. The reduced number of relevant papers in the second phase indicates that our two-phased search strategy was effective and had good evidence coverage.

VII. Conclusion

This systematic literature review framed research works that generated UML components from NL. We focused on approaches used, their pros and cons, and metrics. In addition, we identified the contributions and limitations of these works in this study. We found several heuristic-rule-based solutions. It can be a potential avenue to build a robust framework using machine learning techniques by exploiting these heuristic rules. In many research fields, several common standard datasets exist that help researchers extend their research work

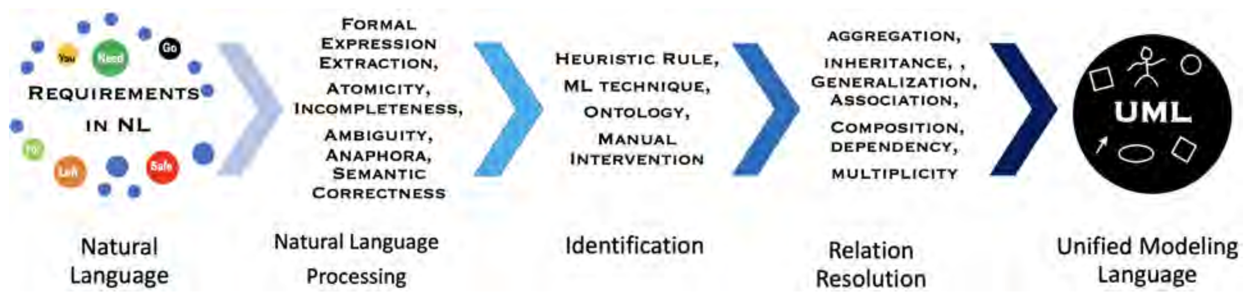

Requirements in NL
You
Need
Go
Fill
Lift
Safe
Formal Expression Extraction,
Atomicity, Incompleteness,
Ambiguity, Anaphora, Semantic Correctness
Heuristic Rule, ML technique, Ontology, Manual Intervention
Aggregation, Inheritance, , Generalization, Association, Composition, Dependency, Multiplicity
UML
Natural Language
Natural Language Processing
Identification
Relation Resolution
Unified Modeling Language

Fig. 3. Conceptual Framework

rapidly. We believe that establishing a benchmark dataset and designing metrics (quantitative and qualitative) for evaluating NL to UML transformation will help the community.

This systematic literature review will help the researchers of the NLP and RE fields. This review pointed out major NL problems that researchers are trying to resolve. Furthermore, we have analyzed and conceptually framed the existing approaches and their pros and cons. Hence, our work will help both the researchers and developers working on the automation of NL to UML transformation.

References

[1] D. Berry, R. GacituaPete, S. Sri, and F. Tjong, "The case for dumb requirements engineering tools," *Int. Working Conf. on Requirements Engineering: Foundation for Soft. Quality REFSQ: Requirements Engineering: Foundation for Soft. Quality*, 2012.

[2] J. Rumbaugh, I. Jacobsen, and G. Booch, "The unified modeling language reference manual." *2d ed., Addison-Wesley*, p. 3, 2004.

[3] E. A. Abdelnabi, A. M. Maatuk, and M. Hagal, "Generating uml class diagram from natural language requirements: A survey of approaches and techniques," *IEEE 1st Int. Maghreb Meeting of the Conf. on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, 2021.

[4] E. A. Abdelnabi, A. M. Maatuk, T. M. Abdelaziz, and S. M. Elakeili, "Generating uml class diagram using nlp techniques and heuristic rules," *20th Int. Conf. on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2020.

[5] N. Zhou and X. Zhou, "Auto-generation of class diagram from free-text functional specifications and domain ontology," *Artificial Intelligence - CiteseerX*, 2004.

[6] N. Bashir, M. Marjani, M. Bilal, N. Malik, M. Liaqat, and M. Ali, "Modeling class diagram using nlp in object-oriented designing," *National Computing Colleges Conf. (NCCC)*, 2021.

[7] A. M. Maatuk and E. A. Abdelnabi, "Generating uml use case and activity diagrams using nlp techniques and heuristics rules," *DATA'21: Int. Conf. on Data Science, E-learning and Info. Systems*, 2021.

[8] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys, Vol.54, No.3, Art. 55.*, 2021.

[9] O. S. Dawood and A.-E.-K. Sahraoui, "From requirements engineering to uml using natural language processing –survey study," *EJERS, European Jour. of Engg. Research and Science Vol. 2, No. 1*, 2017.

[10] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, 08 2004.

[11] M. Soeken, R. Wille, , and R. Drechsler, "Assisted behavior driven development using natural language processing," *Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation TOOLS '12: Objects, Models, Components, Patterns*, 2012.

[12] M. Ibrahim and R. Ahmad, "Class diagram extraction from textual requirements using natural language processing (nlp) techniques," *IEEE 2nd Int. Conf. on Computer Research and Development*, 2010.

[13] D. K. Deeptimahanti and R. Sanyal, "Semi-automatic generation of uml models from natural language requirements," *ISEC: Proc. of the 4th India Soft. Engineering Conf.*, 2011.

[14] A. M. Alashqar, "Automatic generation of uml diagrams from scenario-based user requirements," *Jordanian Jour. of Computers and Info. Tech. (JJCIT), Vol. 07, No. 02*, 2021.

[15] I. Bajwa and M. C. and, "From natural language soft. specifications to uml class models",," *Int. Conf. on Enterprise Inf. Systems (ICEIS), Berlin*, 2011.

[16] R. Sharma, P. Srivastava, and K. Biswas, "From natural language requirements to uml class diagrams,," *AIRE, Ottawa, ON, Canada, 2nd Workshop on Artif. Intell. for Requirements Eng*, 2015.

[17] I. Bajwa, A. Samad, and S. M. and, "Object oriented soware modeling using nlp based knowledge extraction," *Europ. Jour. of Sci. Research, vol. 35(1)*, 2009.

[18] N. Zhou and X. Zhou, "Automatic acquisition of linguistic patterns for conceptual modeling," *INFO 629: Artificial Intell., pp. 1-19*, 2004.

[19] A. Oliveira, N. Seco, and P. G. and, "A cbr approach to text to class diagram translation," *8th European Conf. on Case-Based Reasoning, Turkey*, 2006.

[20] M. Clavel, M. Egea, and V. Silva, "The mova toola rewriting-based uml modeling, measuring, and validation tool," *12th Conf. on Sof. Eng. and Databases, Spain.*, 2007.

[21] D. Popescu, S. Rugaber, N. Medvidovic, and D. B. and, "Reducing ambiguities in requirements specifications via automatically created object-oriented models," *Monterey Workshop, pp. 103-124*, 2008.

[22] H. Krishnan and P. Samuel, "Relative extraction methodology for class diagram generation using dependency graph," *Int. Conf. On Commun. Control & Comp. Tech, pp. 815-820*, 2010.

[23] V. Sharma, S. Sarkar, K. Verma, A. Panayappan, and A. K. and, "Extracting high-level functional design from soft. requirements,," *16th AsiaPacific Soft. Eng. Conf., pp. 35-42*, 2009.

[24] H. Afreen and I. S. B. and, "Generating uml class models from sbvr soft. requirements specifications," *23rd Conf. on Artif. Intell., pp. 23-32*, 2011.

[25] O. S. D. Omer, A.-E.-K. Sahraoui, M. M. E. Mahmoud, and andAbd El-Aziz Babiker, "Requirements and design consistency: A bi-directional traceability and natural language processing assisted approach," *European Jour. of Engg. and Tech. Research Vol 6,Issue 3*, 2021.

[26] O. Dawood and A.-E.-K. Sahraoui, "Toward requirements and design traceability using natural language processing," *European of Engineering and Tech. Research*, pp. 42–49, Jul. 2018.

[27] P. More and R. P. and, "Generating uml diagrams from natural language specifications,," *Jour. of Applied Inf. Sys., vol. 1(8), pp. 19-23*, 2012.

[28] S. Joshi and D. Deshpande, "Textual requirement analysis for uml diagram extraction by using nlp," *Jour. of Comp. Appl., vol. 50(8), pp. 42-46*, 2012.

[29] H. Herchi and W. Abdessalem, "From user requirements to uml class diagram," *Int. Conf. on Comp. Related Knowledge (ICCRK), Tunisia. arxiv*, 2012.

[30] S. Overmyer, B. Lavoie, and O. R. and, "Conceptual modeling through linguistics analysis using lida," *the 23rd Int. Conf. on Soft. Eng. (ICSE), Canada.*, 2001.

[31] H. Harmain and R. G. and, "Cm-builder: A natural language-based case tool," *Jou. of Auto. Soft. Eng., vol. 10(2), pp. 157-181*, 2003.

[32] A. Arellano, E. Carney, and M. A. A. and, "Frameworks for natural language processing of textual requirements," *Natural language processing of textual requirements The 10th Conf. on Sys., Spain*, 2015.

[33] W. B. A. Karaa, Z. B. Azzouz, A. Singh, N. Dey, A. S. Ashour, , and H. B. Ghazala, "Automatic builder of class diagram (abcd): an application of uml generation from functional requirements," *Jour. of Soft. Practice and Experience, vol. 46(11), pp. 1443-1458*, 2015.

[34] R. D. Mathias and S. R. Wille, “Automated and quality-driven requirements engineering,” *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2014.

[35] W. F. Tichy and S. J. Koerner, “Text to soft.: developing tools to close the gaps in soft. engineering,” *FoSER’10, Santa Fe, New Mexico, USA*, 2010.

[36] A. Al-Hroob, A. T. Imam, and R. Al-Heisa, “The use of artificial neural networks for extracting actions and actors from requirements document,” *Info. and Soft. Tech. Vol. 101, pp. 1-15*, 2018.

[37] V. S, S. Aithal, and P. Desai, “An approach towards automation of requirements analysis,” *Proc. of the Int. MultiConf. of Engineers and Computer Scientists Vol I IMECS.*, 2009.

[38] L. Michl and R. Garigliano, “Nl-oops: A requirements analysis tool based on natural language processing,” *WIT Transactions on Info. and Communication Technologies, vol 28*, 2002.

[39] R. Giganto, “Generating class models through controlled requirements,” *IMECS, Hong Kong*, 2009.

[40] S. Vemuri, S. Chala, and M. Fathi, “Automated use case diagram generation from textual user requirement documents,” *IEEE 30th Canadian Conf. on Electrical and Computer Engineering (CCECE)*, 2017.

[41] M. S. O. Z. A. B. J. Alrawashdeh, “Generate use case from the requirements written in a natural language using machine learning,” *IEEE Jordan Int. Joint Conf. on Electrical Engineering and Info. Tech. (JEEIT)*, 2019.

[42] M. Elallaouia, K. Nafilb, and R. Touahnia, “Automatic transformation of user stories into uml use case diagrams using nlp techniques,” *The 8th Int. Conf. on Ambient Systems, Networks and Technologies(ANT)*, 2018.

[43] S. Nasiri, Y. Rhazali, M. Lahmer, and N. Chenfour, “Towards a generation of class diagram from user stories in agile methods,” *Int. Workshop on the Advancements in Model Driven Engineering (AMDE), Warsaw, Poland*, 2020.

[44] D. K. Deeptimahanti and M. A. Babar, “An automated tool for generating uml models from natural language requirements,” *IEEE/ACM Int. Conf. on Automated Soft. Engineering*, 2009.

[45] J. S. Thakur and A. Gupta, “Automatic generation of sequence diagram from use case specification,” *ISEC ’14, Chennai, India*, 2014.

[46] S. Nanduri and S. R. and, “Requirements validation via automated natural language parsing,” *Jour. of Manag. Inf. Sys., vol. 12(3)*, 1995.

[47] J. Börstler, “User-centered requirements engineering in record-an overview,” *Jour. Nordic Workshop on Programming Environment Research (NWPER)*, 1996.

[48] S. Delisle, K. Barker, and I. B. and, “Object-oriented analysis: Getting help from robust computational linguistic tools,” *4th Int. Conf. on Appl. of Natural Language to Inf. Sys., Austria*, 1999.

[49] S. MacDonell, K. Min, and A. Connor, “, autonomous requirements specification processing using natural language processing,,” *arXiv preprint*, 2014.

[50] C. Arora, M. Sabetzadeh, A. Goknil, L. C. Briand, and F. Zimmer, “Narcia: An automated tool for change impact analysis in natural language requirements,” *ESEC/FSE’15, Bergamo, Italy*, 2015.

[51] P. Jain, K. Verma, A. Kass, and R. G. Vasquez, “Automated review of natural language requirements documents: Generating useful warnings with user-extensible glossaries driving a simple state machine,” *ISEC’09, Pune, India*, 2009.

[52] M. W. Anwar, I. Ahsan, F. Azam, and W. H. Butt, “A natural language processing (nlp) framework for embedded systems to automatically extract verification aspects from textual design requirements,” *ICCAE, Sydney, NSW, Australia*, 2020.

[53] M. S. Haris and T. A. Kurniawan, “Automated requirement sentences extraction from soft. requirement specification document,” *SIET ’20, Malang, Indonesia*, 2020.

[54] T. R. Silva and B. Fitzgerald, “Empirical findings on bdd story parsing to support consistency assurance between requirements and artifacts,” *EASE: Evaluation and Assessment in Soft. Engineering*, 2021.

[55] U. S. Shah and D. C. Jinwala, “Resolving ambiguities in natural language soft. requirements: A comprehensive survey,” *ACM SIGSOFT Soft. Engineering Notes Vol. 40 No. 5*, 2015.

[56] K. A. Memon and X. Xiaoling, “Deciphering and analyzing soft. requirements employing the techniques of natural language processing,” *ICMAI’19, April 12–15, Chengdu, China*, 2019.

[57] C. Wangy, F. Pastorey, A. Goknily, L. Briandy, and Z. Iqbal, “Automatic generation of system test cases from use case specifications,” *ISSTA: Proc. of the Int. Symposium on Soft. Testing and Analysis*, 2015.

[58] C. Huertas and R. Juárez-Ramírez, “Nlare, a natural language processing tool for automatic requirements evaluation,” *CUBE: Proc. of the CUBE Int. Info. Tech. Conf.*, 2012.

[59] A. B. Rojas and G. B. Sliesarieva, “Automated detection of language issues affecting accuracy, ambiguity and verifiability in soft. requirements written in natural language,” *Proc. of the NAACL HLT Young Investigators Workshop on Computational Approaches to Languages of the Americas, Los Angeles, California*, 2010.

[60] T. Yue, L. C. Briand, and Y. Labiche, “atoucan: An automated framework to derive uml analysis models from use case models,” *ACM Transactions on Soft. Engineering and Methodology, Vol. 24, No. 3, Art. 13, Pub.*, 2015.

[61] B. R. Bryant, B.-S. Lee, F. Cao, R. R. Raje, A. M. O. W. Zhao, J. G. Gray, and C. C. Burt, “From natural language requirements to executable models of soft. components,” *Monterey Workshop on Soft. Engineering for Embedded Systems, Chicago, IL, pp. 51- 58*, 2003.

[62] A. Schlutter and A. Vogelsang, “Knowledge extraction from natural language requirements into a semantic relation graph,” *IEEE/ACM 42nd Int. Conf. on Soft. Engineering Workshops (ICSEW)*, 2020.

[63] D. P, S. T, and B. R, “Natural language processing-enhanced extraction of sbvr business vocabularies and business rules from uml use case diagrams,” *Data & Knowledge Engineering Vol. 128, 101822*, 2020.

[64] E. S. Btoush and M. M. Hammad, “Generating er diagrams from requirement specifications based on natural language processing,” *Int. Jour. of Database Theory and Application Vol.8, No.2*, 2015.

[65] A. Arellano, E. Carney, and M. A. Austin, “Natural language processing of textual requirements,” *ICONS’15, The 10th Conf. on Sys., Spain*, 2015.

[66] T. C. de Sousa, J. R. Almeida, S. Viana, and J. Pavón, “Automatic analysis of requirements consistency with the b method,” *ACM SIGSOFT Soft. Engineering Notes Vol. 35 No. 2*, 2010.

[67] J. Karpovic and L. Nemuraite, “Transforming sbvr business semantics into web ontology language owl2: Main concepts,” *17th Int. Conf. on Info. and Soft. Technologies IT2011, Lithunia*, 2011.

[68] G. Deshpande, “Sreyantra: automated soft. requirement inter-dependencies elicitation, analysis and learning,” *IEEE/ACM 41st Int. Conf. on Soft. Engineering: Companion Proc. (ICSE Companion)*, 2019.

[69] B. H and H. R, “Natural language generation from class diagrams,” *https://dl.acm.org/doi/pdf/10.1145/2095654.2095666*, 2011.

[70] A. Sree-Kumar, E. Planas, and R. Clarisó, “Extracting soft. product line feature models from natural language specifications,” *SPLC ’18, Gothenburg, Sweden*, 2018.

[71] K. Kolthoff, “Automatic generation of graphical user interface prototypes from unrestricted natural language requirements,” *34th IEEE/ACM Int. Conf. on Automated Soft. Engineering (ASE)*, 2019.

[72] A. Sukys, L. Nemuraite, E. Sinkevicius, and B. Paradauskas, “Querying ontologies on the base of semantics of business vocabulary and business rules,” *17th Int. Conf. on Info. and Soft. Technologies IT2011, Lithunia*, 2011.

[73] D. M. Berry, P. D. C. Science, M. M. Krieger, and P. D. Mathematics, “From contract drafting to soft. specification: Linguistic sources of ambiguity,” *A Handbook Version 1.0*, 2000.

[74] G. Carvalho, D. Falcão, F. Barros, A. Sampaio, A. Mota, L. Motta, and M. Blackburn, “Test case generation from natural language requirements based on scr specifications,” *SAC’13, Coimbra, Portugal.*, 2013.

[75] N. Vitacolonna, “Conceptual design patterns for relational databases,” *17th Int. Conf. on Info. and Soft. Technologies IT2011, Lithunia*, 2011.

[76] U. Iqbal and I. S. Bajwa, “Generating uml activity diagram from sbvr rules,” *6th Int. Conf. on Innovative Computing Tech. (INTECH)*, 2016.

[77] D. D. Kumar and R. Sanyal, “Static uml model generator from analysis of requirements (sugar),” *2008 Advanced Soft. Engineering and Its Applications*, 2008.

[78] G. Lucassen, M. Robeer, and F. Dalpiaz, “Extracting conceptual models from user stories with visual narrator,” *Requirements Eng (2017) 22:339–358*, 2017.

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