PARG: An Ontology-Integrated AI Framework for Process-Aware Requirement Generation

Md. Masudur Rahman, Rownok Jahan Mowmita, Md. Ashadullah Jamil Jim and Umme Jamila
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
Email: {masudur.rahman0413.cse@aust.edu, mowmita878@gmail.com, ajjim566@gmail.com, ummej67@gmail.com}

***Abstract*—Transforming informal stakeholder descriptions into clear and actionable software requirements is a critical challenge in requirements engineering. Existing Artificial Intelligence (AI)-based tools can process natural language but often overlook alignment with underlying business processes and coverage of all relevant scenarios. This paper proposes a hybrid neural-symbolic framework, referred to as Process-Aware Requirement Generation (PARG), which integrates transformer-based process recognition with ontology-guided requirement generation. PARG extracts key process elements, such as actors, actions, conditions, and outcomes from user stories, maps them to domain-specific processes, and instantiates structured requirement statements using predefined templates. The framework further validates consistency and evaluates coverage to ensure completeness and traceability. By combining language understanding with structured process knowledge, PARG reduces manual effort, supports improved traceability, and produces requirements that are coherent, process-aware, and aligned with real-world workflows.**

Keywords*—*Requirements Engineering, User Stories, Natural Language Processing, Ontology-Based Modeling

I. Introduction

Requirements Engineering (RE) is a key phase of software development that transforms stakeholder needs into clear and verifiable system requirements. Effective requirements engineering helps to ensure that the software correctly reflects the intended system behavior, performance requirements, and operational constraints [1]. In agile development, these needs are commonly expressed as user stories written in natural language. Although user stories are easy to understand, their informal structure often introduces ambiguity and inconsistency. As a result, converting them into structured and process-aligned requirements requires significant manual effort and time [2], [3].

The increasing complexity of modern software systems and the rapid adoption of agile methodologies have made requirements engineering more challenging, particularly in ensuring consistency, completeness, and traceability of requirements derived from user stories [4], [5]. In practice, stakeholders often express requirements in informal natural language, which leads to interpretation gaps and incomplete understanding of underlying business processes. This creates a need for automated approaches that not only process textual information but also incorporate domain knowledge to ensure that generated requirements are structurally sound and aligned with real-world workflows.

Recent advances in transformer-based language models, such as BERT and GPT, have encouraged the development of automated techniques for various RE tasks. These models have been applied to activities, such as requirement identification, classification, defect detection, and completeness analysis [6]–[9]. Despite these advancements, most existing approaches primarily focus on analyzing requirement texts and pay limited attention to the underlying business processes that shape system behavior.

Another limitation of current AI-based RE tools is that user stories are often treated as isolated textual fragments rather than as components of a broader business process. Consequently, the generated requirements may be linguistically correct but not always consistent with real-world workflows [8], [10]. Ontology-based approaches have been used to introduce domain knowledge and reduce ambiguity [10], while hybrid or multi-agent techniques have also been explored for requirement elicitation [11], [12]. However, only a limited number of studies integrate language understanding with structured process knowledge.

To address this gap, this paper proposes *PARG: an Ontology-Integrated AI Framework for Process-Aware Requirement Generation*. The paper introduces a conceptual framework that integrates transformer-based process element identification with ontology-guided requirement templates to support the generation of structured requirements from user stories. In addition, the proposed approach outlines a scoring mechanism to assess process alignment and a coverage analysis module to identify potential missing process steps. By combining language understanding with structured process knowledge, PARG aims to support the generation of more structured requirements that are better aligned with real-world workflows. The proposed framework is useful in reducing manual effort in requirement specification and improving the structural consistency and process alignment of user stories in agile development environments.

The proposed framework is currently conceptual and is intended to provide a structured foundation for integrating language models that can guide future research on process-aware and knowledge-driven requirement generation. A prototype implementation and empirical evaluation are planned as part of future work.

The remainder of this paper is organized as follows.

Section II reviews related work on large language models and ontology-based approaches in requirements engineering. Section III presents the proposed PARG framework with its architecture and algorithms. Section IV describes the preprocessing and requirement generation process. Section V provides an illustrative example demonstrating the feasibility of the approach. Finally, Section VI concludes the paper and outlines future research directions.

II. Related Work

Recent studies in requirements engineering have applied artificial intelligence (AI) to reduce manual effort, particularly in transforming stakeholder inputs into structured software requirements. This section briefly reviews work on large language models and ontology-based approaches in this context.

A. LLMs and Generative AI in Requirements Engineering

Large language models (LLMs) have recently been explored as tools for supporting requirements engineering tasks. Prior work has used these models for activities, such as requirement documentation, refinement, and validation [2]. However, Cheng et al. [3] note that most studies focus on analyzing existing requirements rather than generating structured requirements from informal inputs.

Some recent approaches attempt to address this limitation. Habib et al. [6] proposed ReqBrain, a task-specific LLM designed to generate ISO-compliant requirements. Sami et al. [11] introduced a multi-agent framework for generating user stories from stakeholder descriptions. Although these approaches demonstrate the potential of generative models, they mainly emphasize textual quality. As a result, the generated outputs may not always capture the structure of the underlying business processes.

B. NLP and Ontology-Driven Approaches

Natural Language Processing (NLP) has been widely applied in requirements engineering. Earlier studies used NLP techniques for requirement classification, similarity detection, and completeness analysis [13], [14]. These methods help organizing large requirement sets and supporting the review process.

Another research direction focuses on ontology-based models that incorporate domain knowledge into requirements analysis. By representing domain concepts and relationships explicitly, these approaches help reduce ambiguity and improve requirement consistency [10].

However, transformer-based methods still face difficulties with ambiguity and process-level reasoning [8]. Many existing approaches treat requirements as isolated statements rather than parts of a broader workflow.

Overall, prior work shows that NLP, LLMs, and ontology-based techniques can assist several aspects of requirements engineering. However, most approaches remain focused on textual analysis and provide limited integration with process knowledge. Our proposed *PARG* framework addresses this gap by combining transformer-based process recognition with ontology-guided requirement generation to produce structured, process-aware requirements.

III. Proposed Methodology

The proposed methodology presents a process-aware framework for generating structured software requirements from stakeholder user stories. The framework integrates process ontology knowledge with neural language models to better interpret informal descriptions provided by stakeholders. The primary objective is to transform these user stories into clear and well-structured requirements while ensuring consistency with the underlying domain processes.

Let $U = \{u_1, u_2, ..., u_n\}$ denote the set of stakeholder user stories, and let $O = \{p_1, p_2, ..., p_m\}$ represent the process ontology consisting of formally defined process concepts. Each user story is processed through several stages to extract relevant information and generate requirements that align with the concepts defined in the ontology. The overall framework and its corresponding algorithms are described in the following subsections.

A. Framework Overview of PARG

The PARG framework offers a structured approach for converting stakeholder user stories into process-aware software requirements. The workflow combines user stories, domain ontology, and predefined templates to produce validated requirements. As depicted in Fig. 1, the transformation process comprises five stages: Pre-Phase, Preprocessing, Neural-Symbolic Process Recognition, Requirement Generation, and Validation.

Phase 0: Pre-Phase. In this initial step, stakeholder user stories are collected, and the domain ontology is initialized. The ontology defines formal process concepts and relationships, providing structured knowledge to guide process recognition and requirement generation.

Phase 1: Preprocessing. The collected user stories are prepared for analysis through tokenization, normalization, and noise removal. This cleaning ensures the text is standardized and suitable for effective processing by transformer-based NLP models.

Phase 2: Neural-Symbolic Process Recognition. A transformer-based model performs classification and extracts process-relevant elements from the preprocessed user stories. Core components such as actors, actions, conditions, and outcomes are identified and organized for subsequent mapping.

Phase 3: Requirement Generation. The extracted elements are aligned with concepts in the domain ontology. Using predefined templates, these mapped elements are instantiated into structured, traceable requirement statements that reflect the underlying processes.

Phase 4: Validation and Coverage Analysis. Finally, the generated requirements are validated through consistency checks and ontology-based reasoning. A coverage metric is computed to measure how well the requirements represent the processes captured in the ontology.

Figure 1: PARG Workflow

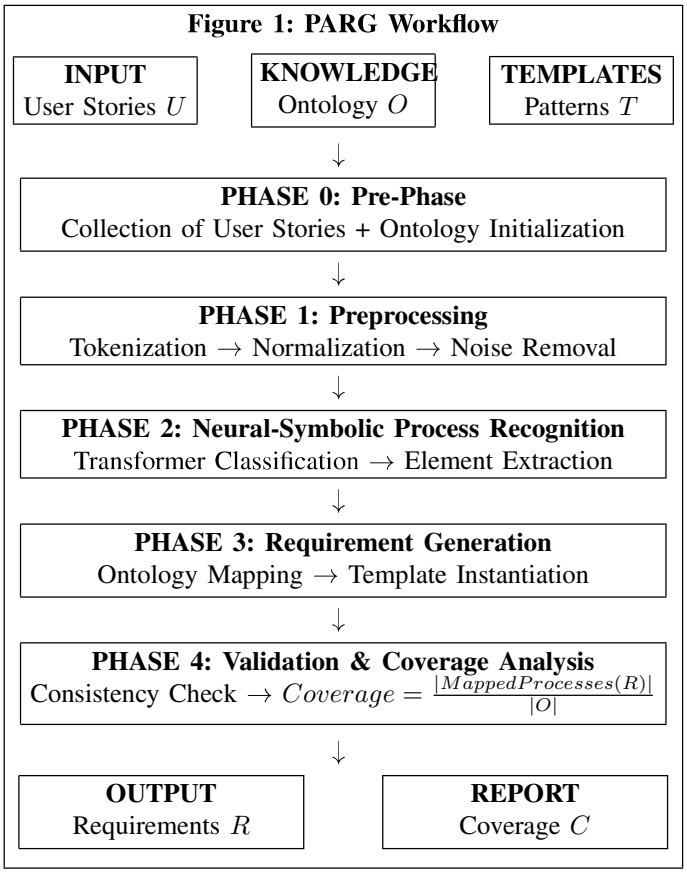

Fig. 1. PARG workflow from user stories to structured process-aware requirements

Algorithm A PARG Framework

Require: User story set U, Process ontology O, Requirement templates T, Confidence threshold θ
Ensure: Structured requirement set R, Coverage report C

Initialize $R \leftarrow \emptyset$
for each user story $u \in U$ **do**
  $u_{clean} \leftarrow$ Preprocess(u)
  $P \leftarrow$ ProcessClassifier(u_{clean})
  $P_{sel} \leftarrow \{p \in P \mid confidence(p) \geq \theta\}$
  $M \leftarrow$ OntologyMapping(P_{sel}, O)
  $ReqSet \leftarrow$ GenerateRequirements(u, P_{sel}, M, T)
  $R \leftarrow R \cup ReqSet$
end for
$C \leftarrow$ CoverageAnalysis(R, O)
Evaluate R using Precision, Recall, F1-score, and BERTScore
if performance $<$ desired threshold **then**
  Update classifier parameters
  Refine ontology relations
  Repeat the process
end if
return R, C

B. Algorithm A: PARG Framework

Explanation: The PARG framework starts by initializing an empty requirement set R. Each user story is preprocessed through tokenization, normalization, and noise removal to produce a clean input. A transformer-based classifier predicts relevant processes, and only processes exceeding the confidence threshold θ are selected. These selected processes are mapped to ontology concepts to ensure domain alignment. Structured requirements are generated using the user story, selected processes, ontology mapping, and templates, then added to R. After processing all user stories, coverage analysis evaluates the representation of ontology processes in R, and the quality of requirements is assessed using Precision, Recall, F1-score, and semantic similarity metrics like BERTScore. If performance is below the desired level, classifier parameters and ontology relations are refined, and the process is repeated until satisfactory results are obtained. The final requirement set and coverage report are then returned.

Algorithm A1 Hybrid Neural-Symbolic Process Scoring

Require: User story u, Process p, Weights α, β
Ensure: Final score $Score(p)$

$C_{model} \leftarrow$ ModelConfidence(u, p)
$S_{ontology} \leftarrow$ OntologySimilarity(u, p)
$Score(p) \leftarrow \alpha \cdot C_{model} + \beta \cdot S_{ontology}$
return $Score(p)$

Algorithm A2 Process Coverage and Risk Detection

Require: Requirement set R, Process ontology O
Ensure: Coverage report C

$covered \leftarrow 0$
$total \leftarrow |O|$
for each process $p \in O$ **do**
  if p is referenced in any requirement in R **then**
    $covered \leftarrow covered + 1$
  else
    Mark p as potential requirement risk
  end if
end for
$Coverage \leftarrow \frac{covered}{total}$
return Coverage report C

C. Algorithm A1: Hybrid Neural-Symbolic Process Scoring

Explanation: This hybrid scoring algorithm computes the final score of a process by combining neural model confidence and ontology-based similarity. First, the confidence from the transformer-based model is calculated for the user story and the process. Next, semantic similarity between the user story and ontology process is computed. The two scores are weighted by α and β and summed to produce a final hybrid score. This score guides the selection of the most relevant processes for requirement generation.

D. Algorithm A2: Process Coverage and Risk Detection

Explanation: The process coverage algorithm evaluates how well the generated requirements represent the ontology processes. For each process in the ontology, the algorithm checks whether it is referenced in the requirement set R. If so, a

counter is incremented; otherwise, the process is marked as a potential gap or risk. Coverage is then computed as the ratio of covered processes to the total number of processes. The resulting report indicates the completeness of the requirement set with respect to the domain ontology.

IV. Preprocessing and Requirement Generation Process

This section describes the detailed processing pipeline of PARG, starting from preprocessing of raw user stories to the generation of structured, process-aware requirements. It also outlines the intermediate analytical steps that transform unstructured text into ontology-aligned requirement representations.

A. Preprocessing of User Stories

Each user story $u \in U$ is first cleaned and prepared to improve the quality of analysis by the language model. This step ensures that the text is consistent and free from noise, making it easier to extract meaningful information.

Preprocessing involves three main steps:

1) **Tokenization:** Splitting the user story into words or linguistic units.
2) **Normalization:** Converting text to a standard form through lowercasing, stemming, and lemmatization.
3) **Noise Removal:** Eliminating stop words, punctuation, and irrelevant symbols.

After these steps, each cleaned user story is converted into contextual embeddings using a transformer-based language model. These embeddings capture the meaning and relationships in the text and serve as input for the next stages.

B. Process Concept Prediction and Requirement Element Extraction

The framework uses the embeddings in two parallel analysis modules:

1) A supervised classifier predicts which process concepts from the ontology are relevant to the user story. Each process $p \in O$ receives a confidence score indicating the likelihood of relevance.
2) A sequence labeling module extracts structured elements from the story, including the actor, action, constraints or conditions, and the intended system outcome.

Together, these extracted elements provide a structured representation that will guide requirement generation.

C. Structured Requirement Generation

Instead of generating a single requirement per user story, PARG produces multiple structured requirements using templates. These templates provide a standard pattern into which extracted elements and predicted processes are inserted.

A semantic mapping function ensures that these elements align with the ontology, keeping the generated requirements consistent with domain knowledge.

D. Validation and Process Coverage Analysis

Once the requirements are generated, they are checked for quality and completeness. Semantic similarity measures help detect duplicates or ambiguous statements, while ontology reasoning ensures that the requirements are logically consistent with the domain processes.

Process coverage measures how well the generated requirements represent the ontology:

$$Coverage = \frac{|MappedProcesses(R)|}{|O|} \tag{1}$$

Here, $MappedProcesses(R)$ is the set of ontology processes referenced in the requirement set R, and $|O|$ is the total number of processes in the ontology. Higher coverage indicates that more domain processes are represented in the requirements.

Overall, PARG provides a clear and structured pipeline for turning user stories into process-aware requirements. It identifies relevant processes using transformer models, aligns them with domain knowledge via ontology mapping, generates structured requirements through templates, and uses scoring and coverage analysis to ensure completeness and consistency.

V. Example of Requirement Generation

User Story: "As a bank customer, I want to transfer money online so that I can pay bills conveniently."

Step 1: Preprocessing Output The user story is first preprocessed through tokenization, normalization, and noise removal. This results in a structured representation capturing key semantic elements such as the actor (bank customer), action (transfer money), and goal (pay bills conveniently).

Step 2: Process Recognition The processed input is then analyzed using a transformer-based process classifier to identify relevant process concepts from the domain ontology. The model predicts a set of candidate processes that are semantically related to the user story.

Step 3: Ontology Alignment and Process Selection The predicted processes are further refined by aligning them with ontology concepts using a hybrid neural-symbolic scoring mechanism. This step helps prioritize the most relevant processes based on both learned representations and domain knowledge.

Step 4: Requirement Generation Using the selected processes and predefined requirement templates, structured requirements are generated. Each requirement is instantiated by mapping extracted user story elements to corresponding ontology concepts, ensuring consistency with domain processes.

Generated Requirements:

- The system shall allow authenticated bank customers to initiate a fund transfer process.
- The system shall verify account validity and balance before executing the transaction.
- The system shall complete the fund transfer process and update account records accordingly.
- The system shall generate a notification upon successful completion of the transaction.

Step 5: Interpretation This example illustrates how PARG transforms an informal user story into structured, process-aware requirements through preprocessing, transformer-based process recognition, ontology alignment, and template-based generation. The resulting requirements reflect the underlying business process and provide a structured representation suitable for further analysis.

VI. Conclusion and Future Work

This paper presented PARG, a hybrid neural-symbolic framework for generating structured, process-aware software requirements from stakeholder user stories. By integrating transformer-based process identification with ontology reasoning and coverage validation, PARG addresses limitations of existing AI-based RE tools that often overlook process structure. Its neural-symbolic scoring mechanism evaluates requirement-process coherence, while template-based generation produces multiple semantically consistent requirements from a single user story. Overall, PARG can reduce manual RE effort, enhance traceability in agile development, and support the creation of process-aligned requirements with minimal human intervention.

Future work will focus on extending the framework's applicability and providing empirical validation through a prototype implementation of PARG. The prototype will integrate a transformer-based model (e.g., BERT) for process classification and sequence labeling with a domain ontology for process representation. The proposed evaluation will follow standard information retrieval and NLP metrics, including Precision, Recall, F1-score, and BERTScore, along with the ontology-based coverage measure defined in the methodology. This evaluation is intended to assess both the quality of generated requirements and their alignment with underlying business processes. Further work includes developing multi-domain datasets, conducting industrial case studies, refining the hybrid scoring and ontology adaptation mechanisms to improve performance across diverse domains, and performing comparative evaluation with existing LLM-based and ontology-based requirement generation approaches.

References

[1] A. Fariha, S. Alwidian, and A. Azim, "A systematic literature review on requirements engineering and maintenance for embedded software," *IEEE Access*, vol. 12, pp. 114263–114279, 2024.

[2] N. Marques, R. Rocha Silva, and J. Bernardino, "Using ChatGPT in Software Requirements Engineering: A Comprehensive Review," *Future Internet*, vol. 16, p. 180, 2024.

[3] H. Cheng, J. H. Husen, Y. Lu, T. Racharak, N. Yoshioka, N. Ubayashi, and H. Washizaki, "Generative AI for Requirements Engineering: A Systematic Literature Review," *Software: Practice and Experience*, vol. 56, pp. 141–170, 2026.

[4] Y. Cheng, A. Kumar, and S. Lee, "Challenges in Automated Requirements Engineering with Natural Language Inputs," *Empirical Software Engineering*, 2026.

[5] D. Marinescu, P. Novak, and M. Ionescu, "Agile Requirements Engineering: Issues of Consistency and Traceability," *IEEE Access*, 2024.

[6] M. K. Habib, D. Graziotin, and S. Wagner, "ReqBrain: Task-Specific Instruction Tuning of LLMs for AI-Assisted Requirements Generation," arXiv preprint arXiv:2505.17632, 2025.

[7] M. A. Zadenoori, J. Dabrowski, W. Alhoshan, L. Zhao, and A. Ferrari, "Large Language Models for Requirements Engineering: A Systematic Literature Review," arXiv preprint arXiv:2509.11446, 2025.

[8] A. M. Rosado da Cruz and E. F. Cruz, "Machine Learning Techniques for Requirements Engineering: A Comprehensive Literature Review," *Software*, vol. 4, p. 14, 2025.

[9] D. Luitel, S. Hassani, and M. Sabetzadeh, "Improving requirements completeness: automated assistance through large language models," *Requirements Engineering*, vol. 29, pp. 73–95, 2024.

[10] C. Antonious and N. Bassiliades, "A tool for requirements engineering using ontologies and boilerplates," *Automated Software Engineering*, vol. 31, p. 5, 2024.

[11] M. A. Sami, M. Waseem, Z. Zhang, Z. Rasheed, K. Systä, and P. Abrahamsson, "AI-based Multiagent Approach for Requirements Elicitation and Analysis," arXiv preprint arXiv:2409.00038, 2024.

[12] M. A. Abbasi, P. Ihantola, T. Mikkonen, and N. Mäkitalo, "Towards Human-AI Synergy in Requirements Engineering: A Framework and Preliminary Study," arXiv preprint arXiv:2510.25016, 2025.

[13] M. Fazelnia, V. Koscinski, S. Herzog, and M. Mirakhorli, "Lessons from the Use of Natural Language Inference in Requirements Engineering Tasks," arXiv preprint arXiv:2405.05135, 2024.

[14] M. Abbas, A. Ferrari, A. Shatnawi, E. Enoiu, M. Saadatmand, and D. Sundmark, "On the relationship between similar requirements and similar software," *Requirements Engineering*, vol. 28, pp. 23–47, 2023.