Towards a Software Reference Architecture for Natural Language Processing Tools in Requirements Engineering

Julian Frattini[1], Quim Motger[2]

[1]Chalmers University of Technology and University of Gothenburg, Sweden
[2]Universitat Politècnica de Catalunya, Spain

Abstract

Natural Language Processing (NLP) tools support requirements engineering (RE) tasks like requirements elicitation, classification, and validation. However, they are often developed from scratch despite functional overlaps, and abandoned after publication. This lack of interoperability and maintenance incurs unnecessary development effort, impedes tool comparison and benchmarking, complicates documentation, and diminishes the long-term sustainability of NLP4RE tools. To address these issues, we postulate a vision to transition from monolithic NLP4RE tools to an ecosystem of reusable, interoperable modules. We outline a research roadmap towards a software reference architecture (SRA) to realize this vision, elaborated following a standard methodological framework for SRA development. As an initial step, we conducted a stakeholder-driven focus group session to elicit generic system requirements for NLP4RE tools. This activity resulted in 36 key system requirements, further motivating the need for a dedicated SRA. Overall, the proposed vision, roadmap, and initial contribution pave the way towards improved development, reuse, and long-term maintenance of NLP4RE tools.

Keywords

software architecture, natural language processing, requirements engineering, focus group

1. Introduction

The prevalence of natural language (NL) in requirements specifications makes natural language processing (NLP) techniques attractive to automate requirements engineering (RE) tasks [1]. This gave rise to the development of NLP4RE tools, which are "*any software (e.g., a script, executable, or web service) that employs NLP technology to support one or more RE activities*" [2]. Examples include tools for requirements extraction [3], traceability link recovery [4], and test case generation [5].

Despite their performance, academic NLP4RE tools exhibit several problems, including a lack of longevity [6], superficial documentation [7], and low reusability in industrial and academic contexts [1]. Many of these problems are connected to the absence of architectural guidance [6], which forces developers of NLP4RE tools to repeatedly re-implement common software steps such as input document parsing, model integration, result visualization, and data formatting for evaluation or benchmarking.

In this paper, we summarize the challenges noted in prior systematic studies (Section 2) and formulate our vision to overcome them (Section 3). In the roadmap to achieve this vision (Section 4) we pick up the suggestion from prior work that a shared software reference architecture (i.e., a high-level blueprint defining common structures, components, and principles to guide system design [8]) can address these challenges [2]. As the first step of this roadmap, we report results from a focus group activity with NLP4RE stakeholders (Section 5), aimed at (1) paving the way towards this vision, (2) supporting its feasibility, and (3) guiding the elicitation of architectural requirements. All materials derived from this study are publicly available at https://github.com/airera/study-sra and archived at https://doi.org/10.5281/zenodo.18268705.

Joint Proceedings of REFSQ-2026 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2026. Poznan, Poland, March 23-26, 2026
julian.frattini@chalmers.se (J. Frattini); joaquim.motger@upc.edu (Q. Motger)
https://julianfrattini.github.io/ (J. Frattini); https://quim-motger.github.io/ (Q. Motger)
0000-0003-3995-6125 (J. Frattini); 0000-0002-4896-7515 (Q. Motger)

2. Background

NLP4RE tools enjoy great popularity [1] yet lack established guidance for their design, development, and maintenance [2]. While the community is adopting standard frameworks for areas such as documentation [7], far less attention has been given to common, canonical guidelines for design and development. Though diverse in their actual implementation, NLP4RE tools share several functional and qualitative aspects. For instance, they share multiple steps of their processing pipeline, such as: reading an input document, pre-processing textual requirement artifacts, analyzing them, and producing some form of output through a user interface [1]. On the other hand, these NLP4RE tools often support similar RE activities (e.g., requirements elicitation or analysis) using similar task types (e.g., classification or tracing & relating) [2]. Furthermore, qualitative aspects such as explainability of results, interoperability of software components and recoverability of experimental results are recurrent concerns across NLP4RE tools [9, 10]. Still, NLP4RE tools are developed in isolation as monoliths, inducing several challenges: (1) Despite functional overlaps, NLP4RE tools **rarely reuse** any existing implementations, forcing the developer to build all parts of the tool, not just the uniquely new parts. (2) NLP4RE tools are mostly academic software, which is often **constrained to very specific use cases** (e.g., only able to read one type of input document). (3) Given the diverse architectures of tools with similar use cases, it is **difficult to compare** them against a common benchmark. (4) Most NLP4RE tools are **not maintained** after publication, making them prone to break soon after their development.

3. Vision

We envision that the landscape of NLP4RE tools would benefit from a shift away from isolated, monolithic tools towards *an ecosystem of interoperable and reusable modules*. This ecosystem would address the aforementioned issues as follows:

1. **Improved reuse**: If interoperable modules encapsulating functionality of common steps of the pipeline of an NLP4RE tool were available, new tools could simply reuse appropriate, existing modules. For example, several NLP4RE tools start by ingesting, formatting, and cleaning a requirements artifact in a *csv* format. If this functionality were encapsulated in an interoperable *input parser* module, all these tools could reuse that same module. This would reduce the development effort to only the new modules and their composition with the reused ones.
2. **Improved applicability**: Interoperable modules would also extend the applicability of NLP4RE tools beyond specific use cases and fixed input-output combinations. For example, an input parser accepting only *csv* files could be replaced with another module accepting *docx* files without modifying the rest of the tool if both parsed the input document to a common internal representation of the requirements.
3. **Improved comparability**: If NLP4RE tools followed a common modular structure, tools addressing the same RE activity and with the same task type could be compared easily on dedicated benchmarks. For example, alternative classification modules could be evaluated in isolation while keeping the rest of the pipeline unchanged.
4. **Improved maintenance**: Shifting from isolated, standalone tools to a common ecosystem of modules would distribute maintenance responsibilities across contributing authors. For example, widely reused modules could benefit from joint maintenance efforts, increasing their longevity.

Figure 1 illustrates this paradigm shift. On the left, tools such as *CiRA* [11] are shown as monolithic systems with highly coupled *input*, *processing logic*, and *output* functionality within a single tool boundary. On the right, these are decomposed into interoperable modules, including *Input Parsers* (e.g., *requirements sentence reader* parsing requirements into a common internal representation), *Preprocessors*, *Processors* (e.g., the *CiRA test case generator*), and *User Interfaces*. This separation enables individual modules to be independently reused, evaluated, maintained, replaced, and composed across NLP4RE tools, while preserving flexibility in tool configuration and evolution.

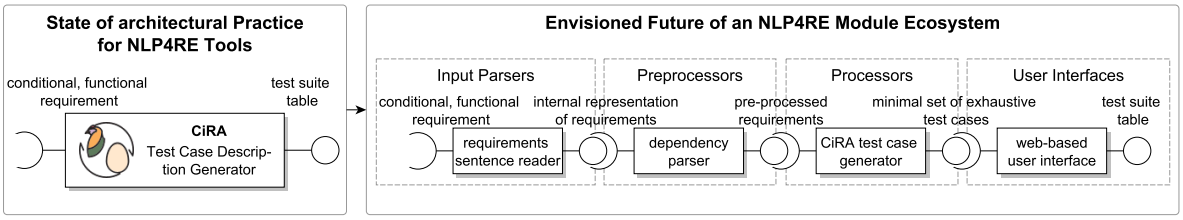

Figure 1: State of practice and envisioned future of NLP4RE tool architectures

4. Roadmap

To achieve the vision outlined in Section 3, we propose the elaboration of a software reference architecture (SRA) as a means to guide the development of interoperable, reusable, and consistent NLP4RE tools. An SRA is an abstraction of domain-specific software architectures guiding and standardizing system design and development practices within that domain [8]. Related work started to address similar problems by building architectural standards and reference models within the NLP4RE community. As an example, Dąbrowski et al. designed a reference model and architecture for review-based user feedback mining within the context of software engineering activities [12]. More recently, Dąbrowski et al. discussed early results on the analysis towards an SRA for agentic RE systems [13]. Despite this, and to the best of our knowledge, no SRAs have been proposed within the context of NLP4RE tools.

To elaborate such SRA, we propose a roadmap following the synthesized guidelines of Nakagawa et al. for designing and evaluating an SRA [14]. The guidelines consist of four steps visualized in Figure 2 and elaborated in Sections 4.1 to 4.4.

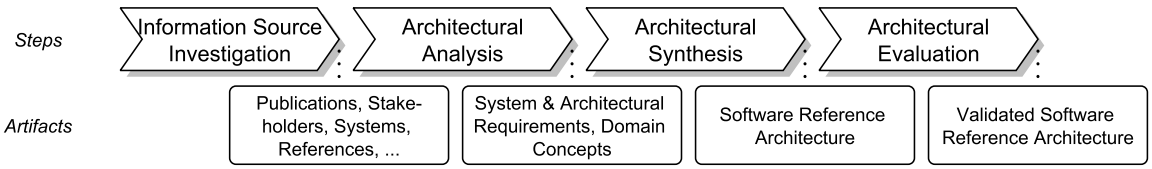

Figure 2: Four step ProSA-RA method for developing SRAs by Nakagawa et al. [14]

4.1. Information Source Investigation

Nakagawa et al. [14] propose to initiate the SRA design by gathering information sources to elicit requirements from. The five types of information map to the domain of NLP4RE tools as follows:

- **Publications.** Systematic mapping studies on articles applying NLP to RE activities, such as information extraction, classification, and traceability [1, 15, 16].
- **Software systems (tools).** Descriptions, classifications and characterization of existing NLP4RE tools that provide insights into architectural decisions and integration practices [2].
- **Reference models and architectures.** Prior reference models and architectural frameworks that serve as conceptual foundations for designing domain-specific NLP4RE SRAs [12, 13].
- **Ontologies.** Domain and requirements ontologies that formalize core concepts, e.g., the OpenReq ontology [17] and NLP4RE-focused taxonomies for documentation of tools [7].
- **Stakeholders.** Experts and practitioners involved in the design, development, or use of NLP4RE tools who contribute their experiences and perspectives.

4.2. Architectural Analysis

From the identified information sources, three types of requirements are to be elicited [14]:

1. **System requirements (SR)**: Key functional and non-functional requirements of existing and envisioned NLP4RE systems.
2. **Architectural requirements (AR)**: A selection and abstraction of the SRs which can be delegated to an SRA. Multiple SRs may map to a single AR when several system-level needs converge into one architectural concern. The resulting set of ARs represents requirements that an SRA must meet such that any NLP4RE tool designed based on the SRA exhibits the selected SRs.
3. **Domain concepts (DC)**: Concepts capturing the main architectural entities and relationships to build a reference model of the NLP4RE domain (containing, for example, data sources, processing components, reasoning layers).

4.3. Architectural Synthesis

Next, Nakagawa et al. suggest to implement an SRA that fulfills the elicited ARs [14]. An SRA shall be specified in terms of architectural views [18], i.e., models of the software architecture from several different perspectives. Nakagawa et al. recommend specifying the architecture from the viewpoint of the design-time modules, runtime components, and deployment on hardware [14]. These views shall be described in diagrams (e.g., UML) and textual explanations to guide developers in the design of their NLP4RE tool. For example, interface specifications would define a common output for all implementations of *input parser* modules to ensure their exchangeability as the initial module of a tool. An open platform to host, share, reuse, and evolve contributions can facilitate the adoption and use of the SRA. Distilling guidelines and structured advice in the form of tutorials or handbooks [2] would further improve the adoption of the SRA.

4.4. Architectural Evaluation

Once developed, the completeness, usefulness, and adoption potential of the SRA feedback needs to be evaluated through expert feedback, tool-based experimentation, and community engagement. For the NLP4RE domain, we envision the following evaluations:

- **Traceability Matrix Mapping**: As done by Dąbrowski et al. [19], mapping architectural modules and components as specified in the SRA to the elements of existing NLP4RE tools shows its general applicability and validates its usefulness by highlighting gaps.
- **Perceived Usefulness**: Surveying NLP4RE tool authors regarding the perceived usefulness, ease of use, and intention to use the SRA validates its applicability.
- **Tool Refactoring**: Refactoring previously published tools according to the SRA will demonstrate its applicability and contribute initial reusable, interoperable modules. First, internal refactoring of existing tools developed by the authors of the SRA [3, 11] will verify that the SRA satisfies the elicited system requirements. Subsequently, external refactoring of third-party NLP4RE tools with community stakeholders will assess whether the SRA meets users' goals in practice.

5. Preliminary Results

To further support our vision and as an initial seed towards architectural requirements elicitation, we contribute to the first step of our research roadmap (Section 4.1) by eliciting information from stakeholders within the NLP4RE community, the only type of information source not readily available (other than publications, systems, etc.). In this section, we summarize the outcomes of a stakeholder-driven focus group activity.

5.1. Method

Aiming at interacting with researchers and practitioners familiarized with NLP4RE tools, we conducted a focus group session at the *12th International Workshop on Artificial Intelligence and Requirements Engineering*[1] (AIRE'25), attended by a sample of our target stakeholders. In a 45-minute activity, 20 workshop participants jointly elicited requirements across four categories: (i) functional requirements, (ii) non-functional requirements, (iii) domain concepts, and (iv) challenges related to the use of generative AI in NLP4RE systems, organized in two iterations.

In the first iteration, participants were divided into four groups, proportionally distributed across categories, each facilitated by a dedicated moderator. Each group was assigned one category and asked to discuss relevant requirements, concepts, or topics, which were captured as sticky notes on a flipboard. To foster prioritization and discussion, participants collaboratively positioned the notes in a two-dimensional matrix ranking *priority* (low to high) and *complexity* (low to high).

After the first iteration, participants from each group—except for the moderator—were redistributed across the remaining flipboards, forming new groups. Moderators then introduced and explained the existing items, after which participants could rearrange notes along the priority and complexity dimensions, and contribute new items. Finally, all participants reconvened, and moderators summarized the outcomes of each category, followed by a brief plenary discussion to clarify interpretations and exchange perspectives. The two authors of this study subsequently consolidated and thematically analyzed the collected information to synthesize and structure the results, serving as the basis for the elicitation of generic system requirements for NLP4RE tools. Our replication package contains additional insights into the instructions provided to participants as well as generated results, including the raw data on the flipcharts and consolidated artifacts such as system-level requirements.

5.2. Results and discussion

The focus group activity resulted in a consolidated set of system-level requirements[2] that directly reflect the four improvement dimensions outlined in our vision (Section 3):

1. **Improved reuse.** Participants emphasized externalizing common pipeline functionality to enable reuse across NLP4RE tools. This is reflected in requirements for modularity and component reuse (SR11), black-box interfaces (SR07), reusable input parsers for common formats (SR02, SR38), and transparent intermediate processing steps (SR06).
2. **Improved applicability.** Stakeholders highlighted the need to overcome narrow, tool-specific use cases. Elicited requirements stress configurability and adaptability, including multilingual support (SR04, SR05), processing of specific RE artifacts (SR03), and explicit communication of input limitations (SR25).
3. **Improved comparability.** Systematic comparison of NLP4RE tools emerged as a key concern. Participants called for evaluation support (SR01), reproducible behavior under fixed conditions (SR17), version tracking (SR18), and durable storage of artifacts and results (SR15), enabling controlled benchmarking of alternative processing modules.
4. **Improved maintenance.** Long-term sustainability was addressed in requirements for modular design (SR10), version control (SR18), robustness and recoverability (SR14, SR15), and legal and data protection compliance (SR08, SR09), supporting shared maintenance of reusable modules.

The elicited requirements provide stakeholder-grounded evidence that a modular, reference-architecture-driven approach can systematically address recurring challenges in NLP4RE tool development.

[1] https://aire-ws.github.io/aire25/

[2] The identifiers *SRxy* refer to system-level requirements as listed in the replication package.

6. Conclusions

In this paper, we articulated a vision for transitioning NLP4RE tools from isolated, monolithic implementations towards an ecosystem of reusable and interoperable modules We motivated this shift through a summary of challenges elicited in prior, systematic studies of the field [1, 6, 2]. We outlined a research roadmap grounded in established SRA development guidelines [14] and provided early empirical support through a stakeholder-driven requirements elicitation activity, resulting in a consolidated set of system-level requirements for NLP4RE tools. Together, these contributions establish a concrete foundation for the design of a dedicated software reference architecture that can guide future tool development and evaluation. As future work, we will derive architectural requirements from the elicited system requirements, assess existing architectures against them, and iteratively design, implement, and evaluate an SRA in close collaboration with the NLP4RE community.

Acknowledgments

We thank the participants of the focus group at the AIRE'25 workshop for their valuable insights.

Declaration on Generative AI

The authors employed generative AI tools, specifically GPT-5.2, for language editing and clarity improvements. All ideas, methods and interpretations remain the sole work of the authors.

References

[1] L. Zhao, et al., Natural language processing for requirements engineering: A systematic mapping study, ACM Computing Surveys 54 (2022) Article 55, 1–41. doi:10.1145/3444689.

[2] J. Frattini, M. Unterkalmsteiner, D. Fucci, D. Mendez, NLP4RE tools: Classification, overview and management, in: A. Ferrari, G. Ginde (Eds.), Handbook on Natural Language Processing for Requirements Engineering, Springer, Cham, 2025. doi:10.1007/978-3-031-73143-3_13.

[3] Q. Motger, et al., Leveraging encoder-only large language models for mobile app review feature extraction, Empirical Software Engineering 30 (2025) 104. doi:10.1007/s10664-025-10660-y.

[4] T. Hey, J. Keim, S. Corallo, Requirements classification for traceability link recovery, in: 2024 IEEE 32nd International Requirements Engineering Conference (RE), IEEE, 2024, pp. 155–167. doi:10.1109/RE59067.2024.00024.

[5] J. Fischbach, et al., Automatic creation of acceptance tests by extracting conditionals from requirements: NLP approach and case study, Journal of Systems and Software 197 (2023) 111549. doi:10.1016/j.jss.2022.111549.

[6] J. Frattini, et al., Requirements quality research artifacts: Recovery, analysis, and management guideline, Journal of Systems and Software (2024) 112120. doi:10.1016/j.jss.2024.112120.

[7] S. Abualhaija, et al., Replication in requirements engineering: The NLP for RE case, ACM Transactions on Software Engineering and Methodology 33 (2024) Article 151, 33 pages. doi:10.1145/3658669.

[8] L. Garcés, et al., Three decades of software reference architectures: A systematic mapping study, Journal of Systems and Software 179 (2021) 111004. doi:10.1016/j.jss.2021.111004.

[9] L. Beqiri, C. S. Montero, A. Cicchetti, A. Kruglyak, Classifying ambiguous requirements: an explainable approach in railway industry, in: 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW), IEEE, 2024, pp. 12–21.

[10] M. Arrabito, A. Fantechi, S. Gnesi, L. Semini, et al., A comparison of nlp tools for re to extract variation points., in: REFSQ Workshops, 2020.

[11] J. Frattini, J. Fischbach, A. Bauer, CiRA: An open-source python package for automated generation of test case descriptions from natural language requirements, in: 2023 IEEE 31st International

Requirements Engineering Conference Workshops (REW), IEEE, 2023, pp. 68–71. doi:10.1109/REW57809.2023.00019.

[12] J. Dąbrowski, et al., Mining user feedback for software engineering: Use cases and reference architecture, in: IEEE 30th International Requirements Engineering Conference (RE), 2022, pp. 114–126. doi:10.1109/RE54965.2022.00017.

[13] J. Dąbrowski, et al., Intelligent agents for requirements engineering: Use, feasibility and evaluation, in: IEEE 33rd International Requirements Engineering Conference (RE), 2025. doi:10.1109/RE63999.2025.00064.

[14] E. Y. Nakagawa, et al., Consolidating a process for the design, representation, and evaluation of reference architectures, in: 2014 IEEE/IFIP Conference on Software Architecture, 2014, pp. 143–152. doi:10.1109/WICSA.2014.25.

[15] M. A. Umar, K. Lano, Advances in automated support for requirements engineering: A systematic literature review, Requirements Engineering 29 (2024) 177–207. doi:10.1007/s00766-023-00411-0.

[16] S.-C. Necula, F. Dumitriu, V. Greavu-Șerban, A systematic literature review on using natural language processing in software requirements engineering, Electronics 13 (2024) 2055. doi:10.3390/electronics13112055.

[17] UH, UPC, OpenReq Ontologies and Patterns Catalogue, Deliverable D5.3, OpenReq Project, European Commission, H2020 Programme, 2018. URL: https://openreq.eu/.

[18] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford, Documenting software architectures: views and beyond, Addison Wesley, 2010.

[19] J. Dąbrowski, et al., Mining user feedback for software engineering: Use cases and reference architecture, in: IEEE 30th International Requirements Engineering Conference (RE), 2022, pp. 114–126. doi:10.1109/RE54965.2022.00017.