Framework to support ageing and obsolescence management using agentic AI

Kweonwoo.sohn*[a], Changhwan.cho[a], Sungjong.kim[a],

[a] *MIRAE-EN Co., Ltd., Disruptive engineering, D403 Daedeok Biz Center, Daejeon, Korea*

**Corresponding author: kwangyoung.sohn@mirae-en.co.kr*

1. Introduction

The adoption of artificial intelligence (AI) in the nuclear sector remains largely confined to pilot projects and limited applications [1]. This is primarily because quality documentation requires not merely readable text, but accurate and reliable data for which errors cannot be tolerated. Such data are typically populated through manual transcription, in which personnel extract and transfer information from specifications, certificates, catalogs, and other source documents with widely varying formats. The nuclear quality assurance field cannot remain outside this technological transition, and further research is therefore needed to support the practical adoption of AI. The ultimate objective of this study is to establish a unified item database from these source materials and, based on the same underlying data, perform not only standardized document generation but also commercial-grade item dedication (CGID), counterfeit, fraudulent and suspect item (CFSI) identification following Part Quality Initiative (PQI), and item equivalency replacement (IER) for the items included in obsolescence and ageing management plan. [2],[3],[4]

2. Scope

This paper focuses on the monorepo[7] configuration designed to support the overall objective described above. Specifically, it addresses the pipeline that parses source materials according to their file formats and converts them into extraction units (RawExtract), the use of JSON Schema to constrain language-model responses to the required structure, and data validation using the Pydantic library. The database schema shared by these components is discussed only in terms of its table structure and relationships, without finalizing the detailed implementation. Template-based document generation, such as for CGID, CFSI, IER will be followed according to this configuration of framework.

3. System Architecture

3.1 System Configuration

As shown in Fig. 1, the proposed system consists of multiple functional layers organized within a single repository. Source materials are first processed according to their file formats and converted into RawExtract units through the extraction pipeline. The extracted data are then passed to the storage layer and stored in the database, from which the document generation component subsequently draws them to populate predefined document templates. These components share a common database schema, enabling the extracted information to be consistently transferred from source document processing to final document generation.

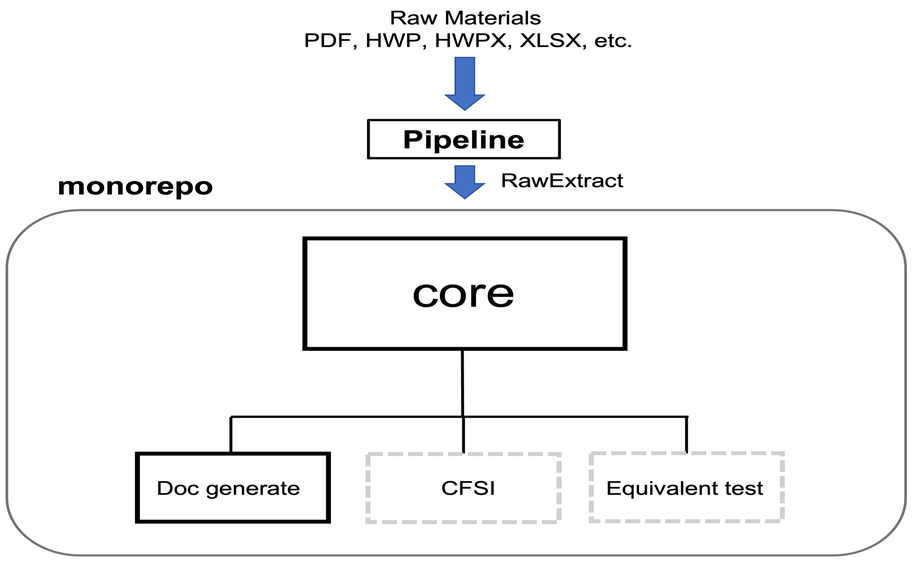


Fig 1. Configuration Diagram

The components are organized within a single repository (monorepo[7]) rather than being separated into individual repositories because all documents produced for reasonable quality assurance for items are based on the same underlying item data.

In other words, the three components generate different type of documents but refer to a common set of data. Accordingly, the item database and its underlying schema should also be shared across the components.

3.2 Pipeline

The pipeline layer handles only the file formats of the source materials and is responsible for preprocessing them into a form that can be interpreted by the language model. A separate adapter is provided for each source file format, and each adapter converts the corresponding source material into an input representation that the language model can read and interpret. The processing tool and segmentation unit for each source file format are summarized in Table I.

Table I Processing Criteria by Source File Format

Source	Processing Tool	Segmentation Unit
pdf(text)	pdfplumber	Row for tables line for non-table text
pdf(Scanned)	pdfplumber(render)	Page
xlsx	openpyxl	Row
csv	Csv+charsetnormalizer	Row
hwpx	zipfile + lxml	Paragraph
hwp	olefile + zlib	Paragraph

3.3 Data Modeling and Validation

The fragment (RawExtract) produced by the pipeline is provided to the language model as input so that the model can interpret its contents. To obtain information in the form required by the user, a JSON Schema such as that shown in Fig. 2 is generated and provided together with the RawExtract. The schema is derived from a Pydantic model in which the expected output structure is defined in advance.[5]

```
{
"properties": {
"name": { "type": "string", "minLength":
1 },
"acceptance_criteria": { "type": "string",
"minLength": 1 },
...
},
"required": ["item_id", "name", "category",
"acceptance_criteria"]
}
```

Fig 2. JSON Schema

Pydantic is a library that converts input values and validates constraints according to fields declared using type hints. It attempts to convert each value in the input string to its declared data type and construct the corresponding object. If any value does not conform to the specified type or constraints, the object is not created and a validation error is raised.

3.4 Application of the Proposed Framework

For the initial implementation of the proposed framework, a manufacturer's catalog provided in PDF format was used as the input, and Fig. 3 illustrates the overall process. The catalog was converted into 400 RawExtract units, which are listed in the left panel. For tabular data, each row was treated as a single RawExtract unit, and the resulting units were subsequently reviewed one by one. The characteristics contained in the selected unit were then extracted under a predefined schema, with the results shown in the right panel. Rather than being represented as free-form text, each characteristic is structured according to predefined fields, including its name, value, characteristic type, and category.

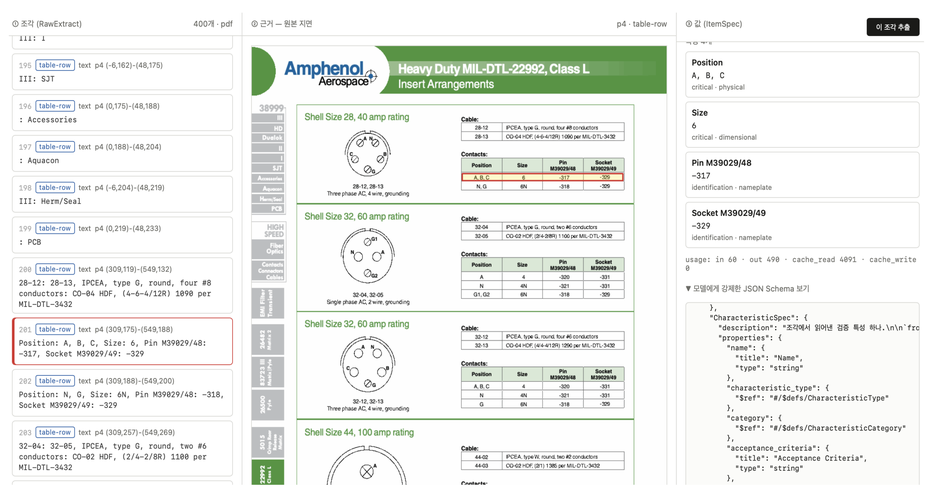


Fig 3. Prototype Validation Interface

4. Conclusion

The significance of this study lies in establishing a minimum requirement for data reliability in quality assurance tasks by preventing language-model outputs from deviating from a predefined structure prior to human review.

Future work will design a database for storing validated values and extend the system to document generation by mapping the stored values to predefined templates as in application layer in Figure 1. Sub-agents will also be developed for plan and/or procedure for CGID, CFSI identification, and IER faced in with(out) CGID, PQI and reverse engineering (RE) to deal with obsolescence and ageing.

Although these sub-agents have different targets of observation, their assessment results will be stored in the same database, thereby maintaining the monorepo architecture throughout future development.

REFERENCES

[1] IAEA, Considerations for Deploying Artificial Intelligence Applications in the Nuclear Power Industry, IAEA Nuclear Energy Series No. NR-T-1.26, STI/PUB/2119, Vienna, 2025.

[2] U.S. Nuclear Regulatory Commission, Definitions, 10 CFR § 21.3(Title 10, Code of Federal Regulations, Part 21).

[3] Electric Power Research Institute, Guideline for the Utilization of Commercial Grade Items in Nuclear Safety Related Applications, EPRI NP-5652, Palo Alto, CA, 1988.

[4] IAEA, Managing Counterfeit and Fraudulent Items in the Nuclear Industry, IAEA Nuclear Energy Series No. NP-T-.26, STI/PUB/1817, Vienna, 2019.

[5] B. T. Willard and R. Louf, "Efficient Guided Generation for Large Language Models," arXiv:2307.09702, 2023.

[6] Anthropic, Tool use (function calling) — structured output via JSON Schema, Anthropic API documentation https://docs.anthropic.com (accessed 2026-08-12).

[7] IAEA, Application of Configuration Management in Nuclear Power Plants, Safety Reports Series No. 65, STI/PUB/1461, Vienna, 2010.