ChatGPT

Recommendations for Improving the ADR Workflow and Base Type System

Context and Key Challenges

The DARF repository uses **Architectural Decision Records (ADRs)** to capture research, design and operational decisions across the platform. The recent re-organization of the `docs` folder aims to integrate a knowledge-graph-backed database and to automate workflows that generate ADRs from documents, logs or multimedia inputs. Each ADR follows a rich metadata schema defined in `docs/process/adr-type-system.md` and uses type codes (e.g., RSH for research, CON for concepts, VEN for vendor choices). Adding new types or workflows currently requires re-evaluating existing ADRs through large LLM calls, which is expensive. The goal is to *improve the scalability and agency* of ADR generation without sacrificing traceability or evidence.

Insights from External Research

1. **Knowledge-driven, multi-agent frameworks enhance architecture design** – The MAAD framework orchestrates specialized agents (analyst, modeler, designer, evaluator) to interpret requirements and generate architecture designs and evaluation reports. It leverages external knowledge through a vector database and retrieval-augmented generation (RAG). Industrial architects found that MAAD's knowledge-driven approach improves correctness and practical usefulness, though human validation remains essential [1].

2. **LLM-based agent systems improve design rationale generation** – A study comparing zero-shot prompting, chain-of-thought (CoT) prompting and multi-agent prompting for design rationale generation found that multi-agent systems produce design rationales more aligned with human experts and contain fewer misleading arguments [2]. However, LLMs still struggle with context-specific details, underscoring the need for domain knowledge and verification [3].

3. **Knowledge graphs strengthen retrieval-augmented generation** – GraphRAG enhances standard RAG by extracting entities, relationships and claims from text to build a knowledge graph. During indexing, documents are segmented, entities and relations are extracted, and hierarchical clustering groups related concepts. Summaries for each community provide rich context [4]. Integrating a knowledge graph improves multi-hop reasoning and retrieval accuracy, addressing the limitations of pure vector-similarity retrieval.

4. **Knowledge-graph-enhanced RAG improves quality and efficiency** – LinkedIn's customer-service QA system constructs a knowledge graph from issue tickets and uses it for retrieval and answer generation. This graph-based method preserves structural relationships between issues and improves retrieval accuracy and answer quality, reducing median resolution time by 28.6 % [5].

5. **Taxonomy adaptation frameworks handle evolving corpora** – TaxoAdapt dynamically expands LLM-generated taxonomies by iteratively classifying documents and extending taxonomy depth and width based on corpus distribution. It generates taxonomies that are 26.51 % more granularity-preserving and 50.41 % more coherent than competitive baselines [6]. The method addresses the challenge of evolving domains and multi-dimensional classification of documents.

6. **Zero-shot and small models are effective for text classification** – Evaluations of LLMs for categorizing scientific texts show that off-the-shelf models like Llama 3 achieve classification accuracy up to 0.82, outperforming traditional BERT models [7]. In requirements engineering, zero-shot T5-XL models achieve the best F-scores for multi-label classification of requirements while using fewer parameters than giant LLMs [8]. These findings suggest that smaller models can perform initial classification cheaply.

7. **Best practices for ADRs stress clarity and maintainability** – The widely used ADR template collection emphasises keeping each ADR focused on one decision, documenting rationale and consequences, and linking related ADRs. ADR names should use imperative verbs and be immutable; new ADRs should supersede rather than edit previous decisions [9].

Recommended Improvements

1. Simplify and Hierarchically Organize the ADR Type System

- **Group the 33 existing types into higher-level categories** (Research, Concept/Technology, Vendor, Agent, Business/Legal, Process, Security, etc.). New subtypes can be added within these categories without re-evaluating the whole corpus.
- **Adopt a dynamic taxonomy adaptation strategy** inspired by TaxoAdapt. Use an iterative classification process where new documents influence the taxonomy's depth and width [6]. Maintain versioned mappings between old and new types to ensure backward compatibility.
- **Make some metadata fields optional or auto-derivable**. Core fields (id, type, date, decision, alternatives, tags) should be mandatory; others (trade-off analysis, linked evidence) can be generated automatically or populated on demand. This reduces manual editing when the type taxonomy evolves.

2. Build a Knowledge-Graph-Backed Ingestion Pipeline

- **Extend** `document-ingestion.sh` **to perform automatic chunking and entity extraction**. Use LLM-based extraction (e.g., GPT-4 or smaller models) to identify entities, relationships and key claims from each document. Store this information in a knowledge graph similar to GraphRAG [4].
- **Leverage the knowledge graph for retrieval**. When generating ADRs, query the graph to retrieve related research, existing decisions, and evidence. This enables multi-hop reasoning and reduces unnecessary LLM calls, as demonstrated by LinkedIn's QA system [5].
- **Automatically link generated ADRs to entities and relationships** in the graph (e.g., technology names, design patterns). This provides traceability and helps visualize decision dependencies.

3. Introduce a Multi-Agent ADR Generation Workflow

- **Decompose the ADR generation process into specialized agents**:

- *Ingestion/Classifier Agent* – uses zero-shot or small models (e.g., T5-XL) to assign incoming documents or logs to the appropriate ADR category with high accuracy [8], reducing reliance on large LLMs.
- *Research Summarizer Agent* – retrieves relevant context from the knowledge graph and summarises key findings. It can also fetch external references when needed.
- *Decision Generator Agent* – drafts the ADR using a larger LLM, guided by prompts and templates. It fills in YAML front matter, context, problem, options and consequences, referencing research.
- *Evaluator Agent* – verifies the ADR for completeness and consistency, identifies missing evidence, and flags issues for human review. This mirrors the MAAD evaluator agent, which helps ensure correctness [1].

- **Use a workflow orchestrator** (e.g., a DAG or the AOV graph model used in the Flow framework) to manage agent execution and dependencies. The orchestrator should support dynamic adjustments – if the evaluator flags missing context, the summarizer can be re-invoked. Modular workflows improve efficiency and allow tasks to run in parallel [10].

- **Incorporate memory**. Store embeddings and outcomes of previous ADR generations so agents can reuse past decisions and avoid redundant processing [11].

4. Employ Efficient Classification to Reduce Large-LLM Calls

- **Use small or medium-sized models for initial classification**. Zero-shot classification models like Llama 3 or T5-XL can categorize documents into ADR types with accuracy comparable to large LLMs [7]. This reduces the number of times expensive models need to be invoked.
- **Implement multi-label classification** when documents span multiple ADR dimensions. For example, a research paper may inform both `RSH` and `CON` ADRs. Multi-label classification frameworks can assign multiple types simultaneously without extra calls [8].
- **Cache classification results** and store them in the knowledge graph, enabling quick reclassification when the type system changes. Re-labelling can then be done by updating mappings rather than re-invoking LLMs.

5. Streamline Templates and Validation Tools

- **Adopt widely-recognized ADR templates** (such as Michael Nygard's or MADR) and align them with your internal schema. Ensure each ADR focuses on a single decision, contains clear rationale and consequences, and references related ADRs [9].
- **Automate template population**. Use the summarizer and decision generator agents to fill in sections based on the extracted context, leaving placeholders only when human expertise is needed.
- **Enhance validation scripts**. Extend `validate-fixed-adrs.sh` to check semantic completeness (e.g., each ADR must reference at least one research source and specify trade-offs) in addition to YAML syntax. Connect the validator to the knowledge graph to verify cross-references.

6. Maintain Human-in-the-Loop Governance

- **Human reviewers should sign off on generated ADRs**. LLM outputs may still contain inaccuracies or lack context [12]. A governance workflow can enforce review before ADRs are merged.

- **Document reevaluation triggers and success metrics**. Record conditions under which ADRs must be re-assessed (e.g., major platform upgrades, new regulatory requirements) to avoid unnecessary re-processing.
- **Use metadata to track confidence and reevaluation deadlines**. The YAML front matter already includes `decision_confidence` and `reevaluate_conditions`; ensure agents populate these fields to guide future reviews.

7. Plan for Incremental Type Evolution

- **Version the ADR type system**. Store each version of `adr-type-system.md` and maintain a mapping table between versions. When a new type is introduced, only reclassify ADRs where the mapping indicates a change.
- **Annotate ADRs with their type version** in the YAML front matter. This facilitates selective migration when the taxonomy evolves.
- **Automate migration scripts** to update front matter fields based on mappings, using the cached embeddings and classification results rather than re-generating content with LLMs.

Expected Benefits

- **Reduced LLM usage and cost** – Leveraging small models for classification and knowledge graphs for retrieval minimizes calls to large LLMs, reducing both compute cost and latency.
- **Scalability and adaptability** – A hierarchical, adaptable taxonomy and modular workflows allow the system to evolve with new document types and user requirements without reprocessing the entire corpus.
- **Improved traceability and reasoning** – Knowledge-graph integration ensures each ADR links to evidence and related decisions, supporting causal analysis and dependency tracing.
- **Enhanced decision quality** – Multi-agent collaboration and human oversight lead to more accurate and trustworthy ADRs, as shown by improvements in multi-agent rationale generation [2].

By adopting these recommendations, the Darf stack's ADR system can evolve from a static, manual process to a dynamic, knowledge-driven workflow that scales with user data, reduces operational costs and maintains agency over architectural decisions.

---

[1] [11] [12] MAAD: Automate Software Architecture Design through Knowledge-Driven Multi-Agent Collaboration
https://arxiv.org/html/2507.21382v1

[2] [3] Using LLMs in Generating Design Rationale for Software Architecture Decisions
https://arxiv.org/html/2504.20781

[4] GraphRAG Explained: Enhancing RAG with Knowledge Graphs | by Zilliz | Medium
https://medium.com/@zilliz_learn/graphrag-explained-enhancing-rag-with-knowledge-graphs-3312065f99e1

[5] Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering
https://arxiv.org/html/2404.17723v1

[6] TaxoAdapt: Aligning LLM-Based Multidimensional Taxonomy Construction to Evolving Research Corpora
https://arxiv.org/html/2506.10737

7 On the Effectiveness of Large Language Models in Automating Categorization of Scientific Texts
https://arxiv.org/html/2502.15745v1

8 Language Models to Support Multi-Label Classification of Industrial Data
https://arxiv.org/html/2504.15922v2

9 GitHub - joelparkerhenderson/architecture-decision-record: Architecture decision record (ADR) examples for software planning, IT leadership, and template documentation
https://github.com/joelparkerhenderson/architecture-decision-record

10 Flow: Modularized Agentic Workflow Automation | OpenReview
https://openreview.net/forum