

Nordic AI Meet 2026

O2. Machine Learning & Intelligent Systems (Wed 10:15-11:15)

2. Graph Representation Learning for Software Architecture Recovery

Authors: Rakhshanda Jabeen^{1,2}; Morgan Ericsson²; Jonas Nordqvist³; Anna Wingkvist²

Affiliations: 1. Electrolux Professional AB, Ljungby, Sweden; 2. Department of Computer Science and Media Technology, Linnaeus University, Växjö, Sweden; 3. Department of Mathematics, Linnaeus University, Växjö, Sweden

Abstract: Software systems evolve continuously, and their architecture documentation is often missing, incomplete, or outdated. This may intensify as AI coding assistants raise code volume without matching documentation. Recovering a medium-sized system's architecture manually is labor-intensive and can take hundreds of expert hours, motivating automated SAR, which infers a system's modular decomposition from its implementation.

Software systems are relational: files are linked by typed static dependencies, organized in folder hierarchies, and carry semantic cues in code identifiers, so SAR suits AI methods that learn from heterogeneous evidence. Many traditional SAR methods rely on hand-crafted heuristics or fixed fusion rules rather than integrating these signals in a data-driven way. Earlier representation-learning work used node2vec over dependency graphs, capturing structure but not code semantics or hierarchy. Recent work applies large language models, but this can be costly at industrial scale, and evaluation on public benchmarks makes it hard to separate genuine reasoning from pretraining exposure.

We study how GNN-based representation learning can improve SAR through three linked questions. First, can learned dependency representations outperform fixed heuristic fusion? Architecture Representation Learning (ARL) trains a heterogeneous graph-attention encoder with a contrastive objective over typed dependencies and adds edge attributes for member-level interactions. Second, does replacing the contrastive objective with reconstruction, while combining all signals, change what the encoder learns? Graph Auto-Encoders for Architecture Recovery (GAER) addresses this with a graph auto-encoder integrating folder hierarchy and code semantics as node features, and compares GCN and GAT encoders. Third, how much does folder organization add beyond dependencies alone, and can diffusion-based rewiring recover similar gains? A controlled variational-GAE study fixes the encoder and varies only the input graph across dependency-only, diffusion-rewired, and folder-augmented variants to isolate each signal.

On open-source Java, C, and C++ systems, ARL and GAER significantly improve over heuristic fusion and node2vec-style baselines, with ARL's largest gains on more homophilic systems, only marginal gains from its edge attributes, and GAT consistently outperforming GCN in GAER. The controlled study shows that recovery from dependencies alone is bounded more by the structural context reaching the encoder than by the encoder itself. Folder structure helps mainly where dependency cohesion is weak, trading dependency-based modularity for closer alignment with the reference decomposition. The results suggest that GNN-based SAR succeeds not merely through a stronger encoder, but by learning to expose and combine complementary forms of architectural evidence. All methods scale to industry-sized systems, finishing in under five minutes on Chromium at roughly 18,000 files.