

Advancing Metadata-Driven Enterprise Architecture Management Tools Using Conceptual Structures

SALEEM, Mahpara <<http://orcid.org/0009-0003-1671-4796>>, POLOVINA, Simon <<http://orcid.org/0000-0003-2961-6207>> and FALLON, Richard <<http://orcid.org/0000-0001-5747-8839>>

Available from Sheffield Hallam University Research Archive (SHURA) at:

<https://shura.shu.ac.uk/37927/>

This document is the Accepted Version [AM]

Citation:

SALEEM, Mahpara, POLOVINA, Simon and FALLON, Richard (2026). Advancing Metadata-Driven Enterprise Architecture Management Tools Using Conceptual Structures. In: Conceptual Knowledge Structures : Third International Joint Conference, CONCEPTS 2026, Montpellier, France, August 31– September 4, 2026, Proceedings. Lecture Notes in Computer Science (16811). Springer Nature Switzerland, 329-339. [Book Section]

Copyright and re-use policy

See <http://shura.shu.ac.uk/information.html>

Advancing Metadata-Driven Enterprise Architecture Management Tools Using Conceptual Structures

Mahpara Saleem¹[0009–0003–1671–4796], Simon Polovina^{2,3}[0000–0003–2961–6207],
and Richard Fallon⁴[0000–0001–5747–8839]

¹ Mahaz Tech Enterprise Limited (MTE), Manchester, UK
`mahpara@mahaztechenterprise.com`

² Department of Computing and Digital Technologies,
Sheffield Hallam University, Sheffield, UK

³ Simile Logics Limited, Sheffield, UK
`S.Polovina@shu.ac.uk`, `simon@similelogics.ltd`

⁴ Red Rooster IT, Vienna, Austria
`richard@fallon.at`

Abstract. Metadata-driven Enterprise Architecture (EA) is critical for aligning business strategy with digital execution, yet metadata often remains fragmented across systems and tools. Moregraph uses Conceptual Graphs as its formal specification and SAP Graph as an exemplar, visualising 4,560 clean-core data objects—but visualisation alone does not ensure governance. This paper extends Moregraph with a formal *Metadata-to-Governance Transformation* (MGT): a principled, deterministic mapping from OData entity type metadata to structured *Data Object* governance records, grounded in five design principles and delivered via two pathways (real-time API and Excel bulk import). Demonstrated using SAP LeanIX across four SAP Graph namespaces, functional validation confirms correctness, completeness, and efficiency gains over manual documentation. The approach generalises to any OData-compliant source and any EA platform supporting programmatic or bulk import.

Keywords: Enterprise Architecture Management · Conceptual Graphs · SAP Graph · SAP LeanIX · Metadata Management · Clean Core · Design Science

1 Introduction

Enterprise Architecture (EA) management aligns an organisation’s technology landscape with its business strategy [3,18]. A persistent challenge is *metadata fragmentation*: data objects and their properties are distributed across disparate documentation, making it difficult for architects to maintain a coherent model of the data landscape [2]. SAP Graph—a unified API layer providing cross-system access to SAP business data objects via OData v4 (a standard, machine-readable format for data entities and relationships) and implementing the SAP Clean

Core strategy [10,11]—exposes 4,560 clean-core data objects programmatically. However, translating this metadata into a governed EA model remains largely manual: at 30–60 minutes per entity, documenting the full clean-core landscape represents weeks of architect effort [6].

Moregraph [8] addresses part of this gap through interactive visualisation. EA governance platforms such as SAP LeanIX [12]—an Enterprise Architecture Management (EAM) tool that stores architecture information as typed *fact sheets*—require structured *Data Object* records capturing identity, properties, relationships, and lifecycle [17,16]. No automated mechanism currently bridges SAP Graph metadata to such records. This paper closes that gap with the formal *Metadata-to-Governance Transformation* (MGT): a deterministic function from OData entity type nodes to EA Data Object governance records, grounded in five design principles. Its novelty lies in (i) formal platform-independent specification, (ii) complete coverage of governance-required fields, and (iii) automated, auditable execution.

The contributions are:

1. A formal, platform-independent MGT mapping Moregraph’s OData-derived metadata graph to structured EA Data Object governance records, grounded in five explicit design principles.
2. Two complementary delivery pathways—real-time API export (Pathway 1) and governance-friendly Excel bulk export (Pathway 2)—demonstrated using SAP LeanIX.
3. Functional validation across four SAP Graph namespaces (five entities) confirming correctness, completeness, consistency, and robust error handling.

2 Background

2.1 SAP Graph, Clean Core, and Design Science

SAP Graph provides consistent access to SAP business data objects via OData v4, spanning SAP S/4HANA (1,479), SAP Sales Cloud (2,027), SAP SuccessFactors (999), and 55 cross-system objects—4,560 clean-core data objects in total [8]. The Clean Core strategy positions SAP Graph as the preferred integration pattern, discouraging direct system modifications [10]. Governing these artefacts within a dedicated EA platform is essential for architectural clarity [2], yet no automated bridge exists. This paper follows Design Science Research (DSR) [1]: the *artefact* is the MGT; the *relevance cycle* established the automation gap; the *design cycle* iterated over transformation rules and pathway designs; the *rigor cycle* grounds the design in OData semantics [5], the LeanIX metamodel [4], and EA governance literature [3].

2.2 Conceptual Structures and EA Alignment

Moregraph’s requirements and conceptual structure were captured in Conceptual Graphs (CGs) [13,14,7]—a formal knowledge representation in which concept

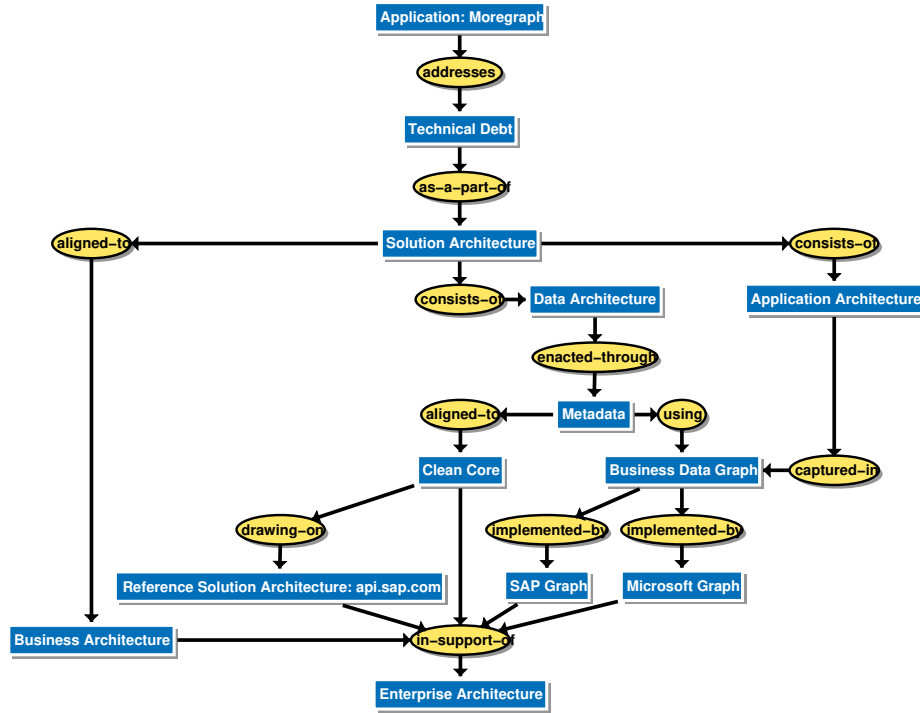


Fig. 1. Prior CG specification of Moregraph [8].

nodes (blue rectangles) denote entities and relation nodes (yellow ovals) denote relationships, designed to be both human-readable and computationally precise. Figure 1 shows Moregraph’s prior CG specification [8], built using CharGer¹, proceeding from [Application: Moregraph] through [Business Data Graph] to [Enterprise Architecture]. Figure 2 enhances this by adding [MGT], formalising how Moregraph transforms SAP Graph metadata into governed [Data Object] fact sheets in [EAM Tool: SAP LeanIX].

In Moregraph’s internal graph, OData entity-type nodes correspond to CG concept nodes and navigation-property edges to CG relation nodes. Table 1 formalises the structural alignment between OData and LeanIX models that grounds the MGT: a *Data Object* in EA practice captures a business data entity’s identifier, properties, relationships, lifecycle status, and provenance [17,16], and SAP LeanIX realises this as a Data Object fact sheet ingested via REST/GraphQL API or Excel bulk import [4].

¹ <https://github.com/Conceptual-Graphs/CharGer>

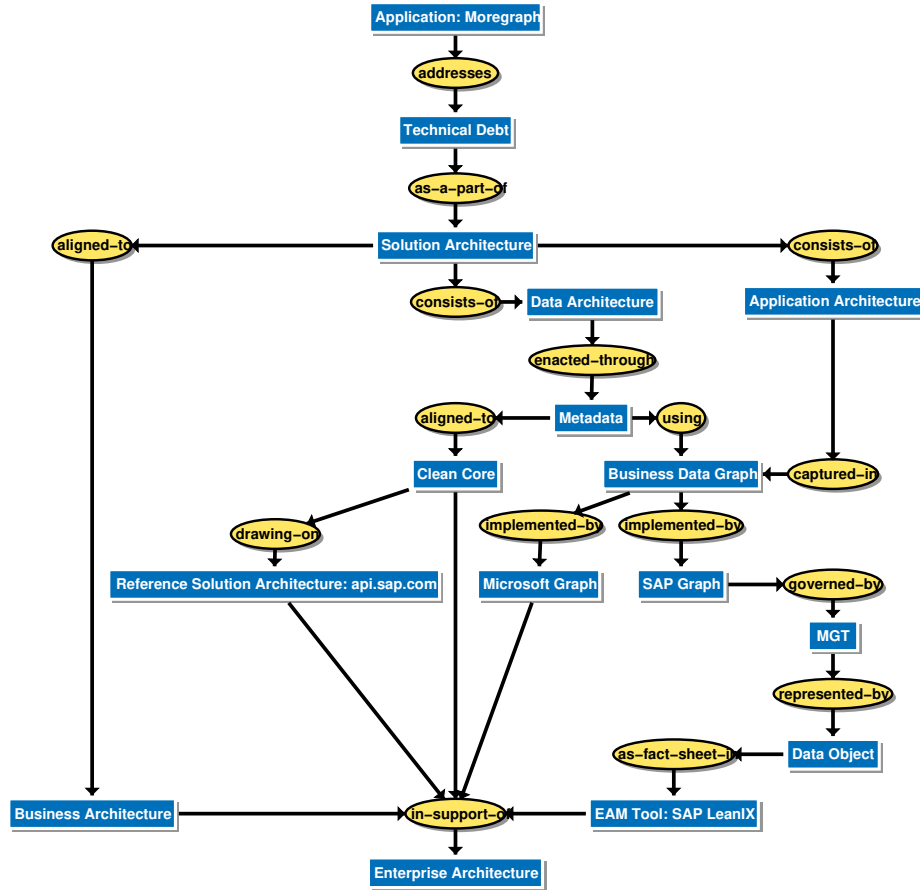


Fig. 2. MGT-enhanced CG specification of Moregraph.

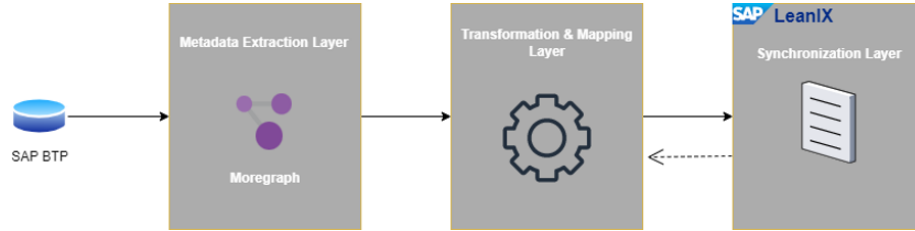
3 The Metadata-to-Governance Transformation

3.1 Research Question and Architecture

Building on Moregraph’s visualisation [8]: *How can Moregraph’s metadata visualisation be integrated with SAP LeanIX to improve EA governance?* The framework has three stages (Fig. 3): (1) **Metadata Extraction**—Moregraph parses OData v4 documents and builds an internal conceptual graph; (2) **Transformation**—the MGT converts graph nodes into LeanIX-compatible structures; (3) **Synchronisation**—Pathway 1 (API) or Pathway 2 (Excel) delivers artefacts to LeanIX.

Table 1. Conceptual alignment between OData v4 entity type and LeanIX Data Object fact sheet models.

OData v4 Entity Type	LeanIX Data Object Fact Sheet
Qualified entity type name	Fact sheet name and external identifier
Property definitions	Custom fields (StringValue typed)
Navigation properties	Relationships to other fact sheets
Namespace / service origin	Provenance description
API documentation URL	Resource link

**Fig. 3.** Three-stage integration architecture (left to right).

3.2 Metadata Extraction

Moregraph parses OData v4 metadata across the `sap.s4`, `sap.c4c`, `sap.hcm`, and `sap.graph` namespaces, building $G = (N, E, \lambda)$: node set N (entity types), directed edges E (navigation properties), and labelling function $\lambda : N \rightarrow A$. For each node $n \in N$:

$$\lambda(n) = \langle id, properties, service, adj(n) \rangle$$

where id is the fully qualified identifier (e.g. `sap.c4c.ServiceRequest`), $properties$ is the list of OData property definitions, $service$ is the system of origin, and $adj(n) = \{m \in N \mid (n, m) \in E\}$ is the set of outgoing navigation-property neighbours.

3.3 Formal MGT Definition

The MGT is $T : N \times M \rightarrow F$, where F is the set of EA Data Object governance records and $M = \{API, Excel\}$:

$$T(n, \mu) = \langle name(n, \mu), type, description, fields, resource \rangle \quad (1)$$

Components are derived deterministically from $\lambda(n)$: $name(n, \mu)$ is $localName(id)$ in API mode (display clarity) or the full id in Excel mode (namespace uniqueness); $type = Data\ Object$; $description = \delta(service, id)$ (template-derived provenance); $fields = \{(p.name, StringValue) \mid p \in properties\}$; $resource = \text{SAP Graph documentation URL}$. For Excel, $T^+(n) = \{T(n, Excel)\} \cup \{T(m, Excel) \mid m \in adj(n)\}$ captures the entity and its outgoing neighbourhood in one batch.

Table 2. Field mapping implementing the MGT (DP = design principle realised).

$\lambda(n)$	Component	LeanIX Field	Mode	DP
$localName(id)$		Name, Display Name	API	DP1
Full id		Name, Display Name	Excel	DP1
Full id		External reference	Both	DP1
$service + id$		Description	Both	DP2
$properties$		Custom fields	Both	DP2
SAP Graph URL		Resource link	API (auto)	DP3
$adj(n)$ (Excel only)		Neighbour rows	Excel	DP5

Five **design principles** ground the MGT: **DP1 Canonical Identity** (fully qualified id as external reference); **DP2 Semantic Fidelity** (auto-populate derivable fields; reserve governance fields for human review); **DP3 Bidirectional Traceability** (resource link to SAP Graph documentation); **DP4 Governance Mode Flexibility** (μ separates transformation from delivery); **DP5 Structural Completeness** (T^+ captures entity and neighbourhood). Table 2 summarises the field mapping.

3.4 Delivery Pathways

Pathway 1 (API) realises $T(n, API)$: (1) obtain OAuth 2.0 bearer token server-side (DP4); (2) POST to `/pathfinder/v1/factSheets` (DP1, DP2); (3) attach SAP Graph documentation link via GraphQL mutation (DP3); (4) return duplicates and API errors as structured inline feedback; (5) navigate user to the new fact sheet on success.

Pathway 2 (Excel) realises $T^+(n)$, generating a workbook formatted to the LeanIX bulk import template. Columns A–H map to: ID (empty for new records), Type (*Data Object*), Name and Display Name (full id , DP1), and governance-reserved fields for Description, Level, Status, and Quality Seal (DP2). The workbook is timestamped for audit traceability (DP3).

4 Case Study and Demonstration

Scenario. A manufacturing organisation runs SAP S/4HANA, SAP Sales Cloud, and SAP SuccessFactors under a clean-core compliance programme with SAP LeanIX as its EA governance platform. Five entities were selected across all four SAP Graph namespaces to evaluate the MGT across the full clean-core landscape.

sap.c4c—ServiceRequest: With 18 outgoing navigation properties, this entity tests both pathways comprehensively. Figure 4 shows the Moregraph graph view with ServiceRequest selected (red outline) and the context menu offering *View on SAP Graph*, *Export to SAP LeanIX* (Pathway 1), and *Export to Excel* (Pathway 2). Pathway 1 ($T(ServiceRequest, API)$) produces a fact

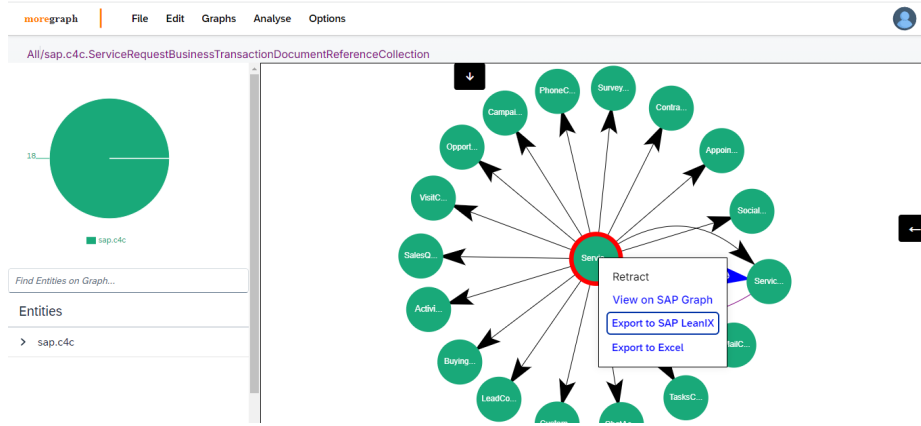


Fig. 4. Moregraph context menu for `sap.c4c.ServiceRequest` (Pathway 1 and 2 options visible).

Table 3. Data Object fact sheets created via Pathway 1 across all four SAP Graph namespaces.

SAP Graph Entity	Namespace Fact Sheet Name	
<code>sap.c4c.ServiceRequest</code>	<code>sap.c4c</code>	ServiceRequest
<code>sap.graph.BusinessPartner</code>	<code>sap.graph</code>	BusinessPartner
<code>sap.s4.A_SalesOrder</code>	<code>sap.s4</code>	A_SalesOrder
<code>sap.hcm.Employee</code>	<code>sap.hcm</code>	Employee
<code>sap.c4c.CorporateAccount</code>	<code>sap.c4c</code>	CorporateAccount

sheet in under five seconds. Pathway 2 ($T^+(ServiceRequest)$) generates a 20-row workbook: one row for ServiceRequest plus 19 rows for its outgoing neighbours, each row populating Name and Display Name with the fully qualified `sap.c4c.*` identifier (DP1) and leaving governance-reserved columns (Description, Level, Status, Quality Seal) blank for EA team enrichment (DP2). The workbook is timestamped at generation for audit traceability (DP3).

sap.s4 and sap.hcm: $T(A_SalesOrder, API)$ and $T(Employee, API)$ each produce fact sheets with namespace-specific provenance descriptions and SAP Graph documentation links, confirming the MGT applies uniformly (Table 3).

Pathway 1 fact sheet: Figure 5 shows the LeanIX fact sheet for `sap.c4c.LeadBusinessTransactionDocumentReferenceCollection`: $localName(id)$ provides the name (DP1); $\delta(service, id)$ the description (DP2); the Resources tab the documentation link (DP3). Status *Check needed* indicates pending governance review (DP2).

Duplicate detection: Re-applying Pathway 1 to *BusinessPartner* already in LeanIX returns a structured inline error with no silent failure. Table 3 summarises Pathway 1 coverage across all four namespaces.

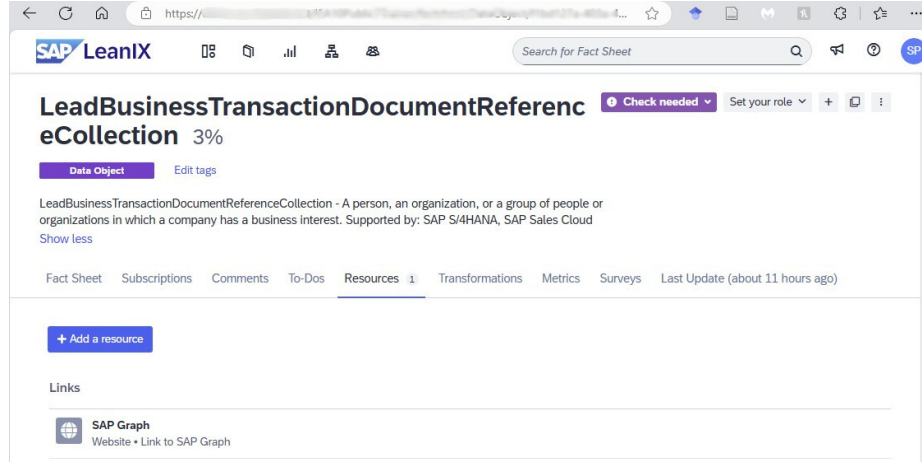


Fig. 5. The `sap.c4c.LeadBusinessTransactionDocumentReferenceCollection` LeanIX Data Object fact sheet for Pathway 1.

Table 4. Functional validation results across five criteria and both pathways.

Criterion	Pathway 1 (API)	Pathway 2 (Excel)	Finding
Correctness	All 5 entities verified	All 20 rows verified	Outputs match $\lambda(n)$
Completeness	All derivable fields set	Full template populated	DP2 fields reserved
Mode consistency	consis- <i>localName(id)</i>	Full <i>id</i> used	Intentional per DP1
Cross-entity	4 namespaces consistent	Batch output consistent	MGT applied uniformly
Error handling	Duplicate surfaced	Import report visible	No silent failures

5 Functional Validation

The validation assessed the MGT against five functional criteria within the DSR evaluation cycle [1]. For each of the five entities, both pathways were applied independently: Pathway 1 outputs were compared field-by-field against $\lambda(n)$ to verify all derivable fields were correctly populated and governance-reserved fields (DP2) intentionally blank; Pathway 2 workbooks were inspected against the LeanIX import template for structural completeness; duplicate detection was tested by re-applying Pathway 1 to an already-imported entity. This is a *functional validation*—an internal artefact assessment—rather than a controlled empirical study; practitioner user studies constitute future work. Table 4 summarises the findings.

As a directional indicator, Pathway 1 produced a complete fact sheet in under five seconds versus 30–60 minutes for manual entry; controlled time-on-task studies constitute future work.

6 Discussion and Related Work

Contribution and Generalisability. The MGT formalises the previously implicit alignment between OData entity types and EA Data Object fact sheets (Table 1), eliminating interpretation variability. DP4’s dual-pathway design accommodates governance models from agile direct API access to formal change-management workflows. Grounding the MGT in OData v4 [5] rather than SAP-specific constructs means its rules apply to any OData v4-compliant source, including Microsoft Graph [8], and DP4’s separation of transformation from delivery enables adaptation to other EA platforms—Ardoq, Essential Project [9], or ServiceNow CMDB—without altering the core mapping. The main limitation is the one-time snapshot: a future incremental synchronisation mechanism monitoring SAP Graph delta changes is the highest-priority extension.

Related Work. CGs [13,7,14] serve as Moregraph’s formal specification (Fig. 1); this paper extends that specification to capture the MGT (Fig. 2). TOGAF [17], Zachman [18,15], and ArchiMate 3.1 [16] provide normative EA structures; Lankhorst et al. [3] survey EA tools comprehensively. Commercial EAM tools such as Ardoq and ServiceNow CMDB support manual metadata entry and limited API population, but lack a formally specified, deterministic transformation from OData v4 sources. Inmon et al. [2] establish metadata governance as prerequisite for EA clarity; Olesen-Bagneux [6] addresses the enterprise data catalogue. Schema.org-based cataloguing and ad-hoc ETL scripts are used for data integration but lack the formal conceptual structure grounding of the MGT. The challenge of bridging OData metadata to EA governance platforms via a formally defined transformation has not been addressed in prior literature.

7 Conclusion

This paper presented the Metadata-to-Governance Transformation (MGT): a formal, platform-independent, deterministic mapping from OData entity-type metadata to structured EA Data Object governance records, grounded in Moregraph’s updated CG specification and five design principles. Two pathways operationalise it: real-time API export (Pathway 1) and governance-friendly Excel export (Pathway 2). Functional validation across four SAP Graph namespaces confirms correctness, completeness, consistency, and robust error handling.

The key insight—that the properties used to describe a data object in metadata are precisely those needed to govern it in an EA platform—is not LeanIX-specific: it holds wherever EA platforms implement the Data Object concept. The framework’s OData-neutral rules generalise to any OData v4-compliant source (e.g. Microsoft Graph), and DP4 enables adaptation to other EA platforms.

Future work will close the synchronisation gap and conduct practitioner user studies.

References

1. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. *MIS Quarterly* **28**(1), 75–105 (2004)
2. Inmon, W.H., O’Neil, B., Fryman, L.: *Business Metadata: Capturing Enterprise Knowledge*. Morgan Kaufmann (2010)
3. Lankhorst, M.: *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Heidelberg, 4th edn. (2017)
4. LeanIX: LeanIX Meta Model. <https://docs-eam.leanix.net/docs/meta-model> (2025), accessed: 28 February 2026
5. OASIS: OData Version 4.0 Part 1: Protocol. <https://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html> (2014), accessed: 28 February 2026
6. Olesen-Bagneux, O.: *The Enterprise Data Catalog*. O’Reilly Media (2023)
7. Polovina, S.: An Introduction to Conceptual Graphs. In: Priss, U., Polovina, S., Hill, R. (eds.) *Conceptual Structures: Knowledge Architectures for Smart Applications*. pp. 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg (2007), <https://shura.shu.ac.uk/1175/>
8. Polovina, S., Fallon, R., Saleem, M.: Moregraph: Metadata-Driven Enterprise Architecture Using Conceptual Structures. In: *Conceptual Knowledge Structures. CONCEPTS 2025*. pp. 91–106. *Lecture Notes in Computer Science*, Springer Nature Switzerland (2025), <https://shura.shu.ac.uk/36156/>
9. Saleem, M., Polovina, S.: Towards Metadata-Driven Enterprise Architecture by Integrating SAP Graph and Essential Project. In: *Proceedings of MOVE2024*. Springer CCIS (2026), in press
10. SAP SE: RISE with SAP—ERP Clean Core Strategy. <https://www.sap.com/products/erp/rise/methodology/clean-core.html> (2024), accessed: 28 February 2026
11. SAP SE: SAP Graph: Unified Business Data Access. <https://api.sap.com/graph> (2024), accessed: 28 February 2026
12. SAP SE: SAP LeanIX Enterprise Architecture Management. <https://www.leanix.net> (2024), accessed: 28 February 2026
13. Sowa, J.F.: *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, Massachusetts (1984)
14. Sowa, J.F.: Conceptual Graphs for Representing Conceptual Structures. In: *Conceptual Structures in Practice*. Chapman and Hall/CRC (2009), num Pages: 36
15. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. *IBM systems journal* **31**(3), 590–616 (1992)
16. The Open Group: ArchiMate 3.1 Specification. The Open Group (2019)
17. The Open Group: TOGAF Standard, Version 10. The Open Group (2025)
18. Zachman, J.A.: A framework for information systems architecture. *IBM Systems Journal* **26**(3), 276–292 (1987)