applied sciences

MDPI

Article

Ontology-Based Validation of Enterprise Architecture Principles

Devid Montecchiari

School of Business, FHNW University of Applied Sciences and Arts Northwestern Switzerland, 4600 Olten, Switzerland; devid.montecchiari@fhnw.ch

Abstract

Enterprise architecture (EA) principles provide normative guidance for architectural evolution, yet validating whether EA models comply with such principles is typically performed manually and does not scale to continuous governance. This paper presents an ontology-based validation approach that enables automated compliance checking of ArchiMate models against EA principles. The approach (i) creates ontology-native representations of ArchiMate models grounded in an enterprise knowledge graph, (ii) structures natural-language principles using SBVR Structured English to reduce ambiguity and support traceability, (iii) enriches the resulting knowledge graph with inferred architectural relations through derivation rules, and (iv) operationalizes validation using SHACL constraints and SPARQL queries that produce explainable violation reports linked to concrete model elements. The approach is developed following Design Science Research and evaluated in three case studies (two real-world organizational settings and one controlled educational setting). The evaluation demonstrates that the approach supports repeatable execution of principle checks on evolving models, improves traceability of violations for architecture review and decision-making, and reduces manual effort by shifting substantial parts of compliance checking from human interpretation to automated constraint validation.

Keywords: enterprise architecture; enterprise architecture principles; ontology; ArchiMate; semantic technologies; knowledge graph; SHACL; SPARQL; automated validation; ArchiMEO; AOAME

1. Introduction

Enterprise Architecture Management (EAM) is widely used to address organizational complexity and continuous change by providing shared descriptions of an enterprise and supporting decision-making in transformation initiatives [1,2]. In practice, EAM relies on established frameworks and modeling languages such as The Open Group Architecture Framework (TOGAF) and ArchiMate to structure architectural descriptions and to communicate architectural decisions across stakeholders [3–5]. While models make architectural knowledge explicit and support communication and consensus, organizations also use enterprise architecture (EA) principles to prescribe how the architecture should evolve in alignment with strategy and governance [2] (p. 20). EA principles therefore have a regulatory role: they justify key design decisions and shape architectural quality by guiding what is acceptable or desirable in the architecture [6,7].

Despite their importance, validating whether an EA model adheres to its stated principles remains largely manual. In many organizations, principle enforcement is embedded in communication-intensive practices (e.g., workshops, reviews, and validation interviews), which are effective for shared understanding but do not scale to continuous, systematic

Academic Editor: Christos Bouras

Received: 1 March 2026
Revised: 19 March 2026
Accepted: 25 March 2026
Published: 30 March 2026

compliance checking [8] (p. 70). This reliance on manual interpretation is also reflected in the broader EAM literature: the benefits attributed to EAM—such as improved decision-making and execution of transformation initiatives—typically depend on ongoing human effort to maintain alignment between business and IT concerns and to ensure governance mechanisms are followed [9–11]. As a consequence, detecting deviations from EA principles in evolving architecture models is time-consuming and error-prone, and it is difficult to achieve repeatable validation across projects, teams, and time.

A core reason is the representational mismatch between principles and models. EA principles are commonly stated in natural language [12], whereas EA models are predominantly graphical. Human stakeholders can interpret both, but direct machine interpretation is hindered because the semantics needed for automated checking are not explicit [13,14]. Prior research has advanced machine-interpretable representations of EA models through enterprise ontologies, semantic lifting/annotation, and ontology-based modeling [15–17]. In this line of work, ontologies can capture both the semantics of a modeling language and domain knowledge, thereby enabling reasoning over model content. Notably, ArchiMEO provides an enterprise ontology that integrates ArchiMate concepts with enterprise semantics [18]. However, what is still missing is a systematic way to extend such formal model representations with formal, machine-interpretable representations of EA principles that can be executed against those models.

The literature on EA principles provides detailed guidance for specifying principles in textual form, including attributes and quality criteria [12,19]. There are also approaches that formalize or analyze principle sets (e.g., identifying conflicts among principles), for example via goal-oriented modeling extensions [20]. Yet these approaches typically do not address the complementary problem that is central for EA governance in practice: validating whether an EA model complies with a set of principles. More generally, the field exhibits diverging directions regarding formalization: principles may be represented informally, semi-formally, or formally [12] (p. 69). Semi-formal approaches such as SBVR are attractive because they support controlled vocabularies and structured rule statements [21,22], whereas semantic web technologies (e.g., RDFS/OWL with rule and constraint mechanisms) provide fully formal representations that support automated reasoning and validation ... [23–26,56]. This raises a practical research challenge: how to exploit the strengths of both worlds—the accessibility of structured principle specifications and the rigor of formal semantic validation—to enable automated checking of EA model compliance.

This paper addresses the following research question: *How can semantic technologies be harnessed to facilitate the automated validation of enterprise architecture models in accordance with enterprise architecture principles?* To answer it, we present an ontology-based validation approach that (i) creates ontology-native representations of ArchiMate models grounded in ArchiMEO, (ii) structures principles using SBVR Structured English to reduce ambiguity and support traceability, (iii) enriches models with inferred relationships via derivation rules, and (iv) operationalizes compliance checking using SHACL constraints and SPARQL queries that produce explainable violation reports linked to concrete model elements. The approach is developed following a Design Science Research methodology and evaluated in three case studies across organizational and educational settings.

This manuscript consolidates and extends earlier work on ontology-based validation of EA principles [27–29]. While prior publications introduced individual building blocks (initial feasibility, a formalization procedure from natural-language principles to executable constraints, and derivation rules for materializing implicit architectural relations), the present paper integrates these elements into a unified and systematically evaluated validation framework.

Taken together, these elements establish the foundation for a systematic and executable validation approach. The contributions of this paper are:

- An end-to-end validation pipeline linking SBVR-structured principles, ontology-grounded ArchiMate models, derivation rule materialization, and SHACL/SPARQL constraint execution;
- A consolidated formalization procedure that preserves traceability from natural-language principle statements to executable constraints;
- A derivation layer that operationalizes implicit ArchiMate relationship semantics to improve robustness of compliance checks;
- A multi-case evaluation demonstrating feasibility and traceable violation reporting across organizational and educational contexts.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 describes the research design and evaluation methodology; Section 4 presents the design of the ontology-based validation approach; Section 5 details its technical implementation; Section 6 presents the validation results from the case studies; Section 7 discusses implications, limitations, and threats to validity; and Section 8 concludes with directions for future work.

2. Related Work

This section reviews prior work in three research streams that underpin automated validation of enterprise architecture (EA) principles: (i) EA principles and compliance practices in EA governance, (ii) ontology-based representations of EA and ArchiMate semantics, and (iii) semantic constraint and rule technologies that enable automated checks. The section concludes by synthesizing the research gap addressed in this paper.

2.1. Enterprise Architecture Principles and Compliance

Enterprise Architecture Management (EAM) evolved from predominantly descriptive, modeling-oriented practices into a strategic management function that supports decision-making, governance, and enterprise transformation [1,3,9,11]. EA models contribute value by facilitating communication, training, analysis, compliance support, and alignment between business and IT stakeholders [10]. In contemporary EAM, these benefits are typically realized through governance structures and processes that institutionalize architectural direction and oversight, such as architecture councils and review boards [30,31].

Within this governance context, EA principles provide normative guidance for architecture design and evolution [3,12]. Principles are commonly described as enduring normative rules that guide how an organization fulfills its mission [3]. A comprehensive definition emphasizes that an architecture principle is a declarative, strategy-based statement that normatively prescribes properties of information system design needed to meet essential requirements [19]. Empirical and survey-based research indicates that principles are widely considered important in practice; however, their specification, operationalization, and measurable impact remain uneven [6,32,33].

A consistent theme across frameworks and practitioner guidance is that principles are primarily captured in textual templates including statement, rationale, and implications, with additional attributes such as preconditions or key actions proposed to improve operational usefulness [3,12,19]. At the same time, principle handling is typically embedded in lifecycle processes that include assessing drivers, formulating principles, stakeholder validation, deployment, and compliance management [34,35]. In practice, compliance management often relies on communication- and review-intensive activities (e.g., workshops, validation interviews, committing reviews), which support shared understanding but remain largely manual and thus difficult to scale for continuous validation [8]. This

tension between normative governance intent and predominantly manual compliance checking motivates research on machine-interpretable representations of principles and automated validation mechanisms.

2.2. Ontology-Based Enterprise Architecture and ArchiMate Semantics

Formalizing EA knowledge has been approached through semantic technologies that aim to provide explicit, machine-interpretable meaning for models and their elements [36,37]. From a modeling-method perspective, semantics is distinct from syntax and notation; automated checks require that semantics be represented in a formal form rather than remaining implicit in diagrams or informal descriptions [38]. In this context, the knowledge space of models can be characterized along the dimensions of use, form, content, and interpretation, where human interpretation differs fundamentally from machine interpretation [13,14].

Ontologies provide a formal and explicit specification of shared conceptualizations, enabling reasoning, classification, and consistency checking [39,40]. In EA settings, enterprise ontologies can represent domain semantics and complement enterprise models, and a range of enterprise ontologies has been proposed [41–44]. Specifically for ArchiMate-centered EA, ArchiMEO demonstrates how modeling-language semantics (the form) and enterprise/domain knowledge (the content) can be integrated to support machine-based interpretation [18]. The feasibility of ontology-based representations for EA and related models has also been illustrated in multiple application contexts [45–47].

Several complementary techniques operationalize semantic enrichment, i.e., the process of making implicit model semantics explicit and machine-interpretable through formal representations. Ontologies provide a formal and shared conceptualization of a domain, enabling logical reasoning, classification, and interoperability across systems [39,40].

A central mechanism in this context is semantic lifting, which transforms model structures into ontology-based representations (e.g., RDF graphs), thereby enabling uniform querying and reasoning over model content [15,48,49]. In contrast, semantic annotation enriches existing model elements by linking them to ontology concepts, improving contextual interpretation and discoverability [16,50]. While semantic lifting focuses on structural transformation, semantic annotation complements it by adding explicit domain meaning.

Ontology-based metamodeling generalizes these ideas by using ontologies themselves as metamodeling languages, replacing traditional meta-modeling approaches (e.g., UML/MOF) with formally defined semantic structures [14,17,51,52]. This approach separates abstract syntax and semantics from concrete notation, enabling models that are cognitively adequate for humans while remaining directly processable by machines. In contrast to post-hoc semantic lifting, ontology-native modeling refers to the direct creation of models as ontology instances, ensuring that semantics are explicitly available from the moment of model creation.

Recent work further emphasizes the value of coherent applied ontologies for enterprise architecture modeling to reduce fragmentation and improve comparability and evaluation [53].

When utilizing ontology-based metamodeling or semantic lifting, some implicit knowledge within the model cannot be directly represented as explicit facts in the ontology. To address this limitation, derivation rules are used to infer additional relationships between elements based on existing model structures. Furthermore, constraint evaluation in this context typically follows a closed-world interpretation, meaning that the absence of required relations or attributes is treated as a violation rather than as unknown information. This assumption aligns with EA governance scenarios, where missing architectural assignments

(e.g., ownership or stewardship relations) are considered findings requiring clarification or remediation.

Despite these advances, the explicit operationalization of EA principles as machine-executable validation logic—particularly in combination with ontology-based ArchiMate representations—remains comparatively underdeveloped.

2.3. Constraint-Based Validation and Semantic Technologies

Automated validation of models can target syntactic correctness, structural or domain constraints, and behavioral properties [38,54]. For EA principle validation, the core challenge is typically not diagram syntax alone, but whether architectural structures and relationships satisfy normative constraints implied by principles. However, principles are typically expressed as natural-language governing statements, and their translation into executable checks requires deliberate formalization choices [20].

The literature offers diverging approaches to principle formalization. Semi-formal rule languages such as RuleSpeak aim to bridge prose and implementation by structuring natural-language rules into practicable statements, yet still require further transformation to become executable [55]. SBVR is commonly highlighted as a promising standard for capturing business vocabulary and rules in a structured, semi-formal form that remains accessible to business stakeholders [12,21,22]. At the same time, semantic web languages provide fully formal representations suitable for machine execution: RDF/RDFS and OWL provide a logic-based foundation, while rule and constraint mechanisms such as SWRL, SPARQL-based approaches, and SHACL enable rule execution, querying, and constraint checking over knowledge graphs [23–26,56].

A key practical bridge between semi-formal SBVR specifications and executable semantic checks is the translation of SBVR statements into OWL-based representations using patterns and transformation rules, which has been explored in multiple studies [57–59]. Related work on ontology-based rule checking demonstrates that semantic rule representations can detect errors and inconsistencies in models and rule sets when domain knowledge is represented explicitly [60]. However, within EA governance, a recurring limitation is that these technologies are often applied either to model representation or to rule formalization, but not integrated into an end-to-end approach that (i) formalizes EA principles, (ii) leverages inferred architectural relations, and (iii) validates compliance directly on an ontology representation of ArchiMate models.

2.4. Summary of Research Gap

In summary, prior work establishes (a) the central governance role of EA principles and the predominance of manual, communication-intensive compliance practices [3,8,34], (b) the feasibility and benefits of representing EA and ArchiMate semantics using ontologies and semantic enrichment techniques [15,17,18], and (c) a mature semantic web technology stack for formal rule execution and constraint validation [24–26,56]. What remains insufficiently addressed is a unified and practically executable mechanism that connects these strands to enable repeatable automated validation of EA principles against ontology-based ArchiMate model representations, including the derivation of implicit architectural relations required for robust principle checking. While these approaches establish the foundations for semantic model representation, their integration with executable validation mechanisms for EA principles remains limited.

3. Materials and Methods

3.1. Research Design

This research follows a Design Science Research (DSR) methodology [61,62]. The objective is to design and evaluate an artifact that enables the automated validation of enterprise architecture (EA) models against enterprise architecture principles using semantic technologies.

The study adopts a constructive and pragmatist research philosophy [63,64], where knowledge is generated through the development and evaluation of an artifact addressing a relevant practical problem. The artifact is an ontology-based validation approach integrating formal representations of EA models and EA principles.

The DSR process was structured into iterative cycles aligned with the following sub-research questions:

- SRQ1: What knowledge is currently machine-interpretable from EA principles?
- SRQ2: What knowledge is currently machine-interpretable from EA models?
- SRQ3: How can the semantics of EA models be represented?
- SRQ4: How can the semantics of EA principles be represented?
- SRQ5: How can EA principles be automatically validated using ontologies?

SRQ1 and SRQ2 correspond to the awareness of the problem phase. SRQ3 and SRQ4 correspond to the design and suggestion phase. SRQ5 corresponds to the development and evaluation phase. Intermediate versions of the artifact (ontology extensions, derivation rules, and validation shapes) were refined based on the findings of successive evaluations. The three case studies were not only used as final evaluation contexts, but also as iterative feedback sources: early cycles focused on feasibility and semantic lifting, subsequent cycles refined principle formalization and derivation, and later cycles stabilized the SHACL/SPARQL validation patterns and reporting.

3.2. Time Horizon

The study adopts a cross-sectional time horizon [65]. Data were collected from organizational contexts at defined points in time, providing a snapshot of existing EA models, EA principles, and validation practices. The objective was not to study longitudinal change, but to assess feasibility and effectiveness of the designed artifact under real-world conditions.

3.3. Case Study Strategy

The artifact was evaluated through three case studies:

- Case 1: FHNW—A real-world university administration context.
- Case 2: Swiss Bank—An anonymized fintech/banking case.
- Case 3: BestCar—A fictional educational case used in classroom settings.

The two real-world cases were used for qualitative evaluation of applicability, feasibility, and stakeholder acceptance. The fictional BestCar case was used for controlled quantitative analysis of principle patterns and validation behavior. Case selection followed three criteria: (i) availability of EA principles in textual form, (ii) availability or feasibility of creating EA models, and (iii) access to stakeholders for validation review. Data collection comprised semi-structured interviews with domain experts and architects (FHNW and Bank cases), review of internal documentation and EA-related artifacts, collaborative modeling sessions to create or refine EA models, and collection of student-generated models and principles in the BestCar case.

EA models were created or consolidated using ArchiMate as modeling language and managed in the AOAME environment, which supports ontology-based metamodeling and semantic annotation. Principles were collected in textual form and subsequently structured for formalization.

3.4. Artifact Development Procedure

Artifact development followed iterative DSR cycles beginning with analysis of machine-interpretable knowledge in EA models and principles (SRQ1–SRQ2), followed by the design of semantic representation mechanisms for models (SRQ3) and formalization procedures for EA principles (SRQ4). The final phase implemented automated validation mechanisms (SRQ5) and applied them in case-based evaluation contexts.

Each iteration involved validation of intermediate results with stakeholders and refinement of modeling conventions and rule specifications.

3.5. Evaluation Protocol

The evaluation protocol consisted of:

1. Collection of EA models and principles.
2. Transformation of models into machine-interpretable representations.
3. Formalization of selected EA principles.
4. Execution of automated validation.
5. Review of validation results with domain experts.

Evaluation focused on (i) feasibility of formalizing principles, (ii) ability to detect principle violations, (iii) clarity and traceability of validation output, (iv) stakeholder confirmation of correctness, and (v) reduction of manual validation effort.

3.6. Reproducibility and Availability of Materials

The study is based on enterprise ontology extensions derived from ArchiMEO, EA models used in the three case studies, formalized principle specifications, and validation rules and queries implementing derivation and compliance logic. Where permitted by organizational confidentiality constraints, anonymized models, ontology extensions, and validation artifacts will be made available upon request. Restrictions related to proprietary organizational data apply to the FHNW and Bank cases.

4. Design of the Ontology-Based Validation Approach

Figure 1 summarizes the conceptual stages of the approach: (i) make EA models machine-interpretable, (ii) reduce ambiguity of principles through structured specification, (iii) enrich model knowledge with derived relations required for reliable checking, and (iv) execute automated validation to produce a traceable report. The following subsections detail these stages and the concrete implementation used in this study.

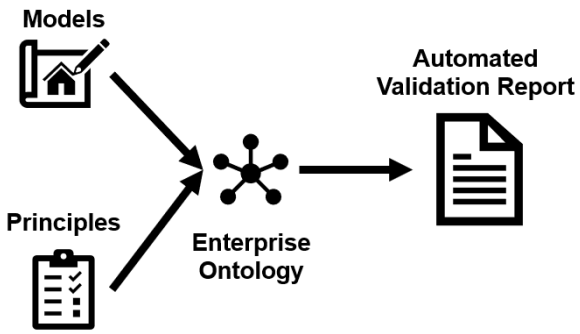

Figure 1. Conceptual overview of the ontology-based validation approach: enterprise architecture models and principles are translated into an enterprise ontology that enables automated compliance checking and reporting.

4.1. Design Objectives and Requirements

The design of the ontology-based validation approach was informed by the findings of SRQ1 and SRQ2, which identified a structural mismatch between enterprise architecture (EA) principles expressed in natural language and EA models represented graphically. While both are interpretable by human stakeholders, neither representation alone provides sufficient semantic explicitness for automated compliance checking.

From this analysis, the following design requirements were derived:

1. Semantic Explicitness of EA Models: EA models must be represented in a machine-interpretable form that preserves modeling language semantics and enables reasoning over architectural elements and relationships.
2. Structured Formalization of EA Principles: EA principles must be transformed from natural-language governance statements into structured and unambiguous representations suitable for machine execution.
3. Derivability of Implicit Architectural Knowledge: Architectural relations that are semantically implied but not explicitly modeled must be derivable to ensure completeness of compliance checking.
4. Traceable and Explainable Validation Outcomes: The validation mechanism must produce structured outputs linking detected violations to concrete model elements, supporting governance transparency and decision-making.

These requirements guided the development of a multi-stage validation approach integrating ontology-based model representation, structured principle formalization, derivation rules for semantic completion, and constraint-based automated validation.

4.2. Ontology-Based Representation of EA Models

Automated validation requires that enterprise architecture (EA) models expose their semantics explicitly and in a machine-interpretable form. Rather than relying solely on post-hoc semantic lifting, this approach adopts ontology-based metamodeling to ensure that model elements are ontology-native from the moment of creation.

ArchiMate was selected as the reference modeling language due to its widespread adoption in enterprise architecture practice. Its meta-model semantics are grounded in ArchiMEO, which integrates:

- Modeling-language semantics (form);
- Enterprise/domain vocabulary (content).

This integration ensures that both syntactic correctness and domain meaning are represented within a unified enterprise knowledge graph. Where necessary, ArchiMEO was extended to reflect updates in the ArchiMate specification and to incorporate additional relations required for principle validation.

The ontology used in this study follows a layered design combining reusable and extensible components. ArchiMEO provides the semantic foundation for representing ArchiMate concepts and relationships. This foundation is extended through the Ontology for Modeling Environment (O4ME), which enables ontology-based metamodeling and model management in AOAME. Finally, enterprise-specific extensions introduce domain concepts and governance-related properties required for principle validation.

The ontology therefore integrates modeling-language semantics and enterprise/domain knowledge within a unified knowledge graph, enabling reasoning, derivation, and constraint validation. A detailed description of the concrete ontology elements and extensions used in the implementation is provided in Section 5.

4.3. Formalization Procedure for Enterprise Architecture Principles

Enterprise architecture (EA) principles are typically formulated as natural-language governance statements. To enable automated validation, these statements must be systematically transformed into machine-executable constraints (extending [28]). Based on the iterative Design Science Research cycles (SRQ4 and SRQ5), this study consolidates the formalization into a structured, multi-stage procedure.

Step 1: Principle Specification and Structuring.

EA principles are first specified using established templates (e.g., statement, rationale, implications). The focus of formalization lies on the normative core expressed in the statement and its operational implications. Ambiguities and implicit assumptions are identified and resolved in collaboration with domain experts.

Step 2: Transformation into SBVR Structured English.

The principle statement is rewritten as a practicable rule using SBVR Structured English. This step introduces explicit classification of:

- *Terms* (noun concepts representing architectural elements);
- *Fact types* (verb concepts representing relations or attributes);
- *Quantifications and modalities* (e.g., "every", "at most one", "it is mandatory that").

This structured representation reduces linguistic ambiguity while preserving stakeholder readability and traceability.

Step 3: Vocabulary Extraction and Ontology Alignment.

Significant SBVR terms and fact types are mapped to ontology constructs. Noun concepts are aligned with ontology classes or individuals, verb concepts with object or datatype properties, and quantifications with cardinality or constraint patterns. Where required, the enterprise ontology (grounded in ArchiMEO) is extended to represent missing domain semantics.

Step 4: Integration of Domain Knowledge and Modeling Conventions.

To ensure that the validation logic can operate on EA models, domain-specific ontologies and modeling conventions are integrated. This includes:

- Extending the enterprise ontology with relevant domain concepts;
- Defining consistent annotation patterns for ArchiMate model elements;
- Ensuring that model elements referenced by principles are explicitly represented and semantically annotated.

This step establishes the semantic preconditions for reliable validation.

Step 5: Encoding as Executable Constraints.

The structured and ontology-aligned rule is encoded as an executable constraint. In this study, SHACL is used as the primary validation mechanism. Node shapes define the target architectural elements, while property constraints and SPARQL-based expressions operationalize cardinality, dependency, and governance conditions. Where inferencing is required, derivation rules are applied prior to constraint evaluation.

Step 6: Validation and Traceability.

The encoded constraints are executed over the enriched enterprise knowledge graph. Validation results are returned as structured SHACL reports linking violations to concrete model individuals and relations. This ensures traceability from natural-language principle to executable rule and back to affected architecture elements.

This multi-stage formalization procedure bridges natural-language governance statements and machine-executable validation logic. By separating structuring (SBVR), semantic alignment (ontology grounding), and execution (SHACL/SPARQL), the approach preserves interpretability while enabling automated compliance checking.

4.4. Derivation and Validation Mechanism

The derivation layer complements the ontology-based representation by materializing architectural relations that are semantically implied but not explicitly modeled. This ensures that validation operates on a semantically complete representation of the architecture rather than on incomplete diagrammatic structures.

Based on a systematic analysis of ArchiMate relationship semantics (e.g., transitivity, structural weakening, dependency propagation), derivation rules are formalized as executable SPARQL CONSTRUCT queries that enrich the knowledge graph with inferred relations.

This enrichment enables validation to consider both explicitly modeled and semantically implied architectural dependencies, reducing false negatives caused by incomplete modeling. At the same time, the separation between derivation and validation preserves transparency, as inferred knowledge is made explicit prior to constraint evaluation.

Validation then proceeds in three steps:

1. Enriching the semantic model with inferred relationships;
2. Executing formalized principle constraints against the enriched model;
3. Generating a validation report listing violations and explanatory traces.

This two-stage mechanism (derivation and validation) ensures both completeness of the analyzed architecture and traceability of validation results.

5. Technical Implementation and Instantiation

This section describes the technical realization of the ontology-based validation approach. While the design in Section 4 is conceptually technology-independent, the evaluation in this study required an executable pipeline that operationalizes (i) ontology-native model creation, (ii) ontology grounding and enterprise extensions, (iii) derivation rule materialization, and (iv) constraint-based validation with explainable reporting.

5.1. Semantic Pipeline Overview

The implementation follows the pipeline shown in Figure 2. Enterprise architecture models and principle specifications are represented as RDF/OWL knowledge graphs, enriched through derivation rules, and validated using SHACL constraints complemented by SPARQL-based expressions. The output is a structured validation report that links each violation to the concrete model elements involved.

The pipeline makes explicit the transformation from model and principle input through ontology grounding, semantic enrichment, and constraint execution to traceable validation output.

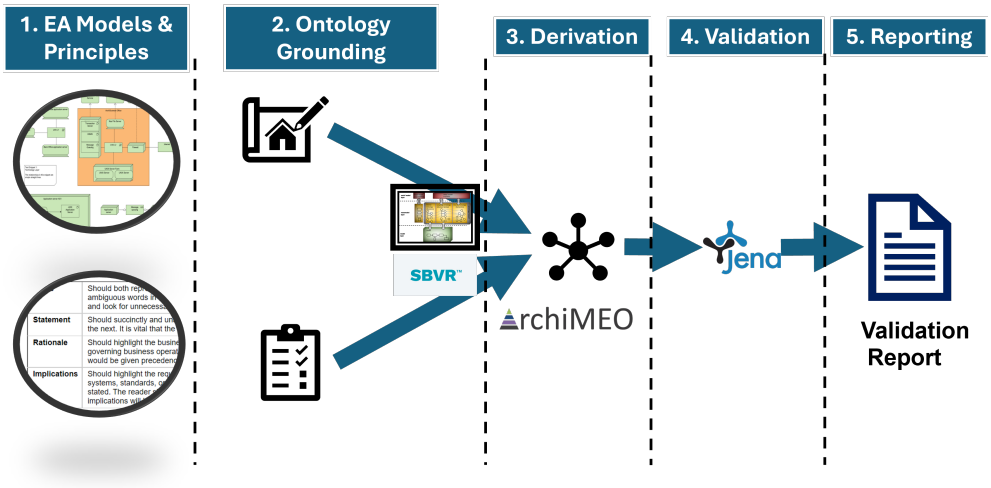

Figure 2. Sequential workflow of the ontology-based validation approach. The process starts from enterprise architecture models and principle specifications, which are transformed into ontology-grounded representations. These are enriched through derivation rules and validated. The workflow produces structured validation reports that link detected violations to concrete model elements, enabling traceable and repeatable compliance checking.

5.2. Ontology-Native Modeling with AOAME

Enterprise architecture models were created and managed using the Agile Ontology-Aided Modelling Environment (AOAME), which implements the Ontology-Based Metamodeling Approach (OBMMA). In AOAME, every modeling element instantiated in the graphical canvas is directly created as an ontology individual. Consequently, the abstract syntax of the modeling language (e.g., ArchiMate concepts), the graphical notation, and the domain semantics are maintained in linked ontologies and remain synchronized during modeling.

Unlike conventional modeling tools, where semantic lifting is performed as a separate export step, AOAME ensures that models are ontology-native from the moment of creation. As a result, rule execution and validation operate directly on the enterprise knowledge graph without requiring an additional transformation layer.

The overall structure of ontology-based metamodeling and validation in AOAME is illustrated in Figure 3. The figure shows how ArchiMate models are instantiated as ontology individuals and integrated with modeling-language semantics, domain knowledge, and derivation rules and validation constraints within a unified enterprise knowledge graph. It also distinguishes ontology-native modeling from optional semantic annotations and illustrates how derivation and validation operate on the same knowledge graph.

The figure also makes explicit the core ontology elements (classes and relationships) and their interconnections across notation, abstract syntax, and domain semantics layers, providing a structural view of how entities and relations are represented and linked within the enterprise knowledge graph.

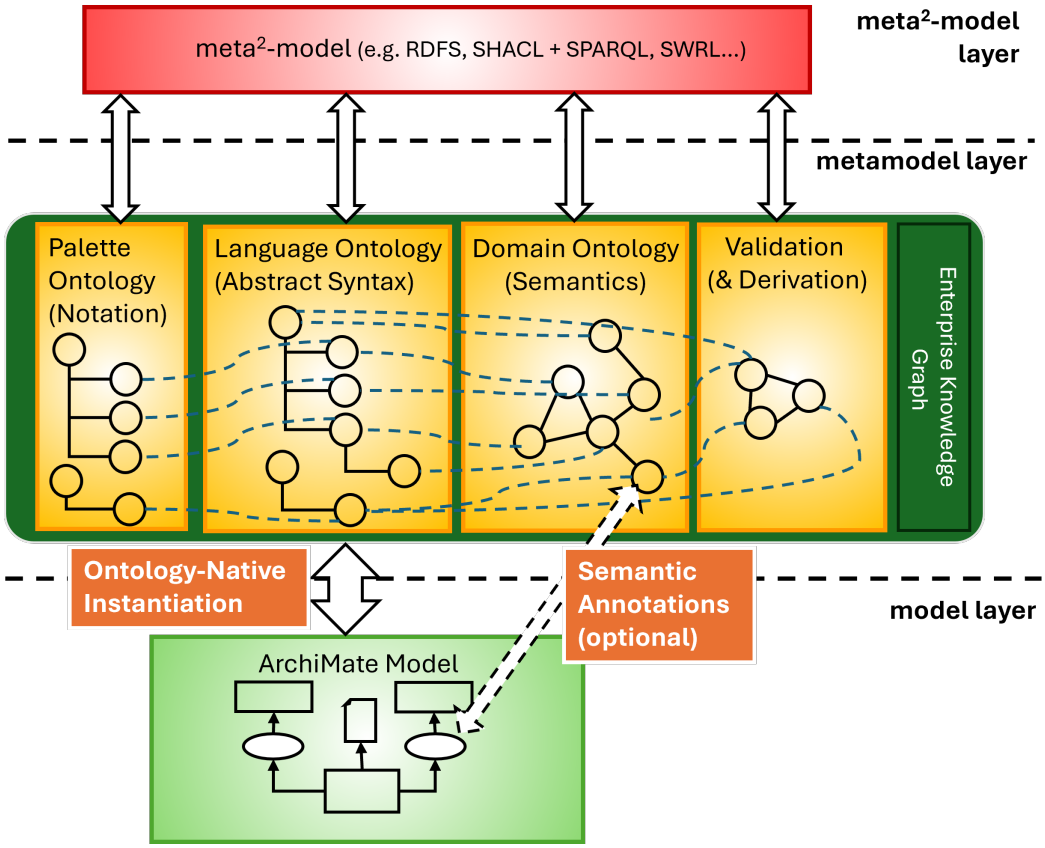

Figure 3. Ontology-based metamodeling and validation structure in AOAME. The figure shows how ArchiMate models are instantiated as ontology individuals and linked to notation, abstract syntax, domain semantics, and derivation rules and validation constraints within a unified enterprise knowledge graph. Adapted from [17,52].

5.3. Ontology Grounding and Enterprise Extensions

The ontology used in this study was constructed through a layered process consisting of (i) reuse of the ArchiMEO ontology, (ii) extension through O4ME for ontology-based metamodeling, and (iii) addition of enterprise-specific domain and governance concepts required for validation.

At its core, the approach reuses ArchiMEO as the semantic foundation for representing the ArchiMate meta-model and its integration with enterprise semantics. ArchiMEO contributes the core architectural element and relationship types used in this study, including classes such as `archi:BusinessProcess`, `archi:ApplicationComponent`, and `archi:DataObject`, as well as ArchiMate relationship concepts such as `archi:Serve` and `archi:Realization`. In this sense, ArchiMEO provides the modeling-language semantics required to represent the form of enterprise architecture models in a machine-interpretable way.

To support ontology-native modeling in AOAME, this semantic base is complemented by the Ontology for Modeling Environment (O4ME), which extends the ArchiMEO stack with constructs required for ontology-based metamodeling and model management [52]. In particular, O4ME introduces concepts such as `mod:ConceptualElement` and linking properties such as `lo:modelingRelationHasSourceModelingElement`, `lo:modelingRelationHasTargetModelingElement`, and `lo:elementIsMappedWithDOConcept`. These constructs enable the explicit representation of modeled elements, their graphical instantiations, and their semantic mappings within a unified knowledge graph. Recent work further positions AOAME as an enterprise knowledge graph editor that supports domain experts in creating and maintaining ontology-grounded models [66].

On top of these reusable layers, the study introduces enterprise/domain extensions required by the evaluated principles. These extensions include organization-specific domain concepts, governance-related constructs such as stewardship roles and master-data classifications, and validation-relevant properties such as usage and ownership mappings (e.g., `eo:AppIsUsedFor`). Through these additions, the ontology captures not only the ArchiMate meta-model but also the enterprise-specific content required for principle validation.

Overall, the resulting ontology integrates (i) reused ArchiMate semantics from ArchiMEO, (ii) ontology-based metamodeling constructs from O4ME, and (iii) case-specific domain and governance extensions. This layered grounding enables validation rules to operate directly on a semantically coherent enterprise knowledge graph that preserves both modeling-language structure and enterprise-specific meaning, which is essential for robust and explainable compliance checking.

5.4. Derivation Rule Execution

Enterprise architecture principles often depend on architectural relations that are implicit in ArchiMate models. Based on a systematic semantic analysis of the ArchiMate specification, derivation rules were formalized to materialize such implicit relations prior to validation. The rules cover transitivity, structural relationship weakening, and interactions between structural, dependency, dynamic, and flow relationships.

The derivation logic was implemented using SPARQL CONSTRUCT queries executed over the RDF graph. Derived relations are stored in the knowledge graph without altering the visual model, preserving readability while enabling machine-complete reasoning. Potential derivation rules suggested by the ArchiMate specification were identified but not always automatically applied, as their validity depends on modeling context and intended interpretation.

5.5. Constraint-Based Validation and Explainable Reporting

Following the derivation phase, principles structured via SBVR and aligned to ontology vocabulary are encoded as SHACL shapes and SPARQL constraints. Validation is executed under a closed-world interpretation, ensuring that compliance is evaluated based on explicitly modeled and derivable knowledge. This interpretation matches governance scenarios where missing assignments (e.g., missing stewardship links) are considered findings requiring clarification or remediation.

The pipeline was executed using Apache Jena for RDF storage, SPARQL processing, derivation rule execution, and SHACL validation. The output consists of structured validation reports in which each violation is linked to the focus node (the element under validation), the violated constraint, and the contributing values or related nodes. This provides traceability back to concrete ArchiMate elements in AOAME and supports governance review and corrective action.

5.6. End-to-End Validation Example

The following simplified example illustrates how the proposed validation pipeline is instantiated end-to-end, from SBVR-based rule structuring to derivation-enabled constraint execution. While the example emphasizes the role of derivation in enabling validation over implicit modeling structures, it follows the six-step formalization procedure described in Section 4.3, condensing intermediate steps. The focus is on (i) SBVR-based rule structuring, (ii) ontology grounding of the rule vocabulary, and (iii) derivation-enabled validation. A comprehensive formalization of derivation patterns is presented in [29].

Principle statement (informal).

Consider a redundancy-avoidance principle stating that applications supporting the same usage context should not redundantly serve the same business process.

SBVR rule structuring (cf. Step 2 in Section 4.3).

The normative core of the principle is expressed in SBVR Structured English to make its logical structure explicit:

Listing 1: SBVR Structured English representation of the principle statement

```
It is obligatory that each Business Process
is served by at most one Application Component
for each Application Usage.
```

This formulation distinguishes:

- **Noun concepts:** Business Process, Application Component, Application Usage;
- **Verb concepts:**
  - *Application Component* is used for *Application Usage*;
  - *Application Component* serves *Business Process*.
- **Modality:** obligatory (a deontic constraint).

Ontology alignment (cf. Steps 3–4).

The SBVR vocabulary is mapped to ontology constructs and modeling conventions:

- *Business Process* → `archi:BusinessProcess`
- *Application Component* → `archi:ApplicationComponent`
- *Application Usage* → domain concept (e.g., APQC classification)
- *is used for* → `eo:AppIsUsedFor`
- *serves* → `archi:Serve` (represented via AOAME relation individuals)

In AOAME, each element is instantiated as a `mod:ConceptualElement` typed with its ArchiMate class and linked to domain semantics via `lo:elementIsMappedWithDOConcept` or domain-specific properties. This enables direct execution of SBVR-grounded rules on the ontology representation without post-hoc semantic lifting.

Derivation as a prerequisite for validation.

In many models, serving relations are not explicitly represented but occur through intermediate elements (e.g., Application Component → Application Service → Business Process). Without enrichment, such cases would lead to false negatives.

To address this, derivation rules materialize implied relations by analyzing relationship chains. The enriched graph thus contains both explicit and inferred knowledge, forming the basis for reliable validation.

From derivation to executable validation.

Listing 2 shows a simplified SPARQL CONSTRUCT rule deriving an implicit `archi:Serve` relation:

Listing 2. Simplified SPARQL CONSTRUCT rule deriving an implicit serving relation.

```
CONSTRUCT {
  ?app archi:serves ?process .
}
WHERE {
  ?realization a archi:Realization ;
      lo:modelingRelationHasSourceModelingElement ?app ;
```

```
      lo:modelingRelationHasTargetModelingElement ?service .

  ?serveRel a archi:Serve ;
      lo:modelingRelationHasSourceModelingElement ?service ;
      lo:modelingRelationHasTargetModelingElement ?process .

  ?app a archi:ApplicationComponent .
  ?service a archi:ApplicationService .
  ?process a archi:BusinessProcess .
}
```

This rule derives a direct `archi:serves` relation by propagating realization and serving links via intermediate application services, making implicit architectural dependencies explicitly available for validation.

Based on explicit and derived relations, the principle is evaluated using SHACL. Listing 3 shows a simplified constraint detecting redundant application support:

Listing 3. Simplified SHACL-SPARQL constraint for redundancy avoidance.

```
ex:RedundancyAvoidanceShape
    a sh:NodeShape ;
    sh:targetClass archi:BusinessProcess ;
    sh:sparql [
        sh:message "A business process is served by more than
        one application component for the same application usage." ;
        sh:select """
            SELECT $this
            WHERE {
              $this a archi:BusinessProcess .
              ?app1 a archi:ApplicationComponent ;
                    archi:serves $this ;
                    eo:AppIsUsedFor ?usage .
              ?app2 a archi:ApplicationComponent ;
                    archi:serves $this ;
                    eo:AppIsUsedFor ?usage .
              FILTER(?app1 != ?app2)
            }
        """ ;
    ] .
```

The constraint detects violations when two distinct application components serve the same business process under an identical usage classification.

After derivation, the SBVR-grounded rule is executed as a SHACL constraint over the enriched graph. This ensures that both explicitly modeled and implicitly derived architectural relations are considered, preventing false negatives caused by incomplete or abstracted modeling structures, while preserving direct traceability to ArchiMate elements in AOAME.

This ensures that validation results are both machine-detectable and explainable, while maintaining direct traceability to the corresponding ArchiMate elements in AOAME.

6. Results

This section reports the results of applying the ontology-based validation approach to three case studies (FHNW, Swiss Bank, BestCar). For each case, we report (i) the principle(s) selected, (ii) the machine-executable formalization (SHACL/SPARQL), and (iii) the validation outcomes produced by the constraint engine, including traceability to violating model elements.

6.1. Overview of Validated Principles and Mechanisms

Table 1 summarizes the evaluated principles, the case contexts, and the primary validation mechanism used. Across cases, principles were formalized into SHACL shapes complemented by SPARQL queries or rules, executed over the ontology representation of ArchiMate models enriched with derived relations.

Table 1. Validated principles and validation mechanisms.

Principle	Case	Validation Mechanism
Unique Master Data Management (MDM) system	FHNW	SHACL constraints with SPARQL-based cardinality checks over ontology annotations.
MDM governance for critical business objects	FHNW	SHACL constraints for required stewardship/governance links.
Redundancy avoidance (no duplicate applications/functions)	Swiss Bank	SHACL constraints with SPARQL-based clustering of domain concepts supported by multiple applications.
Single authoritative update source for data objects	Swiss Bank; BestCar	SHACL constraints with SPARQL-based checks ensuring one owning/updating application per data object.

Across all cases, violations are returned as SHACL validation results that point to the involved focus nodes and contributing model elements, enabling traceability back to the ArchiMate model in AOAME.

6.2. Case 1 (FHNW): Master Data Management and Governance

In the FHNW case, the approach was applied to an ArchiMate model capturing administrative processes and information objects. The selected governance principles focused on ensuring architectural consistency of master data management structures and the presence of required governance constructs.

Unique MDM system validation.

The MDM principle was formalized as a cardinality constraint requiring exactly one application component annotated as the master data management system. Validation was executed on the ontology representation of the model and returned a result indicating whether the architecture contained a single authoritative MDM component. The constraint output provided direct references to the involved application components, allowing stakeholders to verify whether the detected configuration reflected the intended architecture.

MDM governance validation.

The governance principle was validated by checking whether critical business objects were linked to required stewardship and master-data constructs defined in the ontology extension. The SHACL validation identified missing or incomplete governance links for selected objects. Similar to the MDM check, results were returned as traceable constraint violations, enabling the architecture team to review gaps and refine governance assignments.

Overall, stakeholders confirmed that the generated violations were understandable and actionable because each result contained (i) the violated condition, (ii) the specific offending individuals, and (iii) a navigation path back to the original model context.

Each violation record includes the focus node (e.g., an Application Component or Data Object), the violating condition, and the contributing model elements, enabling direct navigation to the originating ArchiMate elements in AOAME.

6.3. Case 2 (Swiss Bank): Redundancy and Single Authoritative Data Updates

In the Swiss Bank case, two complementary principles were validated. First, the redundancy avoidance principle was instantiated as a duplication-detection check: if multiple applications implemented the same domain concept without architectural justification, this indicated potential redundancy. The validation identified several redundancy clusters where the same domain concept was supported by more than one application; these clusters were reviewed with stakeholders as candidates for consolidation or justified exceptions.

Second, a data governance principle required that each data object be updated by only one application component. This constraint operationalizes a single-authoritative-update pattern comparable to a single source of truth interpretation at the operational level. Validation outputs grouped results by data object and listed the responsible applications, allowing architects to quickly identify conflicting update responsibilities.

The output groups results by domain concept or data object and lists the associated application components for each cluster, providing an actionable shortlist for architecture review.

6.4. Case 3 (BestCar): Controlled Validation Behavior Across Many Models

The BestCar case served as a controlled educational setting with multiple independently created ArchiMate models and principle sets. Here, the single-authoritative-update pattern was reused to assess how well the formalization generalizes across heterogeneous model structures and naming conventions. The validation logic remained stable, while model-specific mappings (e.g., term alignment to ontology concepts) required lightweight adjustments. Across student models, the approach consistently produced explainable validation outcomes (compliance or violations) and highlighted where missing modeling links prevented automated checking, thereby making model quality issues explicit.

6.5. Cross-Case Findings

Across the three cases, the results show that the approach enables:

- **Repeatable execution:** once formalized, the same principle checks can be re-run on evolving models without re-interpreting natural-language statements.
- **Traceable outcomes:** SHACL reports link violations directly to model individuals, supporting governance review and corrective actions.
- **Practical feasibility:** principles of different types (data governance and application redundancy) could be expressed as constraints over the ontology representation of ArchiMate models, leveraging inferred relations where needed.

Beyond individual validation outcomes, the cases demonstrate a structured transformation pipeline from natural-language principles to executable constraints. Principles were first structured using SBVR to reduce ambiguity and identify significant terms and fact types. These structured specifications were then aligned with ontology concepts and encoded as SHACL node shapes and SPARQL-based constraints. Derivation rules were applied prior to validation to materialize implicit architectural relations required for reliable checking. This two-stage mechanism (semantic enrichment through derivation followed by constraint-based validation) ensured that both explicit and inferred architectural knowledge were considered during compliance evaluation.

These findings directly address the evaluation criteria defined in Section 3, demonstrating feasibility of principle formalization, reliable violation detection, traceable validation output, and stakeholder-confirmed correctness across contexts. These results motivate the discussion of implications, limitations, and threats to validity in Section 7.

7. Discussion

This study addressed the challenge of reducing manual effort and improving repeatability in enterprise architecture (EA) principle compliance checking by making both sides of the compliance problem machine-interpretable: (i) EA models through ontology-grounded ArchiMate representations, and (ii) EA principles through a traceable transformation from natural language to executable constraints. Across three case studies, the proposed approach operationalized principles as repeatable validation checks and produced explainable violation reports linked to concrete model elements.

A core design choice is the separation of concerns between specification, semantic grounding, semantic completion, and constraint execution. In terms familiar from business rule engineering, EA principles can be seen as governing statements that require refinement into practicable statements and, ultimately, implementation statements. In this work, SBVR Structured English plays the role of a practicable statement: it reduces ambiguity, forces explicit vocabulary and fact types, and preserves stakeholder readability. Executability is then achieved at the implementation level through SHACL/SPARQL constraints grounded in an enterprise ontology (ArchiMEO plus domain extensions). This layered structuring was critical to keep the formalization both feasible in practice and traceable back to the original principle phrasing.

7.1. Interpretation of Findings

Across all cases, validation outputs were considered actionable because they point to (i) the focus node under validation, (ii) the violated condition, and (iii) the contributing values and related nodes. This supports a governance workflow in which automated checking does not replace architectural judgement, but provides a repeatable screening mechanism and a transparent starting point for review and exception handling.

A recurring empirical observation is that *derivation* is a precondition for robust validation. Several violations (or non-violations) depend on relations that are semantically implied by the ArchiMate specification but not consistently modeled explicitly across views (e.g., indirect serving relations via services, propagated dependencies, or transitive effects). Materializing such implied relations prior to constraint execution reduced false negatives and improved robustness across heterogeneous modeling styles. Importantly, the approach keeps derivation and validation conceptually separate: derived knowledge is made explicit in the knowledge graph to enable checks, while validation remains a constraint evaluation step that produces a report.

7.2. Positioning with Respect to Related Work

Prior research has shown the value of semantic lifting and semantic annotation for machine interpretation of EA models, and the semantic web stack provides mature mechanisms for querying and constraint checking. What is comparatively less mature in the EA governance literature is an integrated, end-to-end approach that connects (i) a structured, stakeholder-readable principle representation, (ii) explicit ontology grounding of the principle vocabulary, (iii) derivation of implicit architectural relations needed for reliable checking, and (iv) execution that yields explainable reports linked to model elements. The contribution of this work is therefore not the introduction of SHACL or SPARQL in isolation, but a validation pipeline that preserves traceability from textual principles to executable checks and back to the affected architecture elements.

7.3. Implications for EA Practice

The results suggest three practical implications for organizations that maintain principle sets and ArchiMate models:

Repeatable compliance checking on evolving models: Once encoded, a principle can be executed repeatedly as models change, reducing reliance on ad-hoc, communication-intensive reviews for detecting basic deviations.

Auditability and governance transparency: SHACL validation reports provide a structured and explainable output that can be used in architecture review boards or design checkpoints, because violations are linked to the concrete model individuals involved.

Feedback on model quality and conventions: Automated checks make missing semantic links visible (e.g., missing annotations or missing governance relations). In this way, validation also functions as a quality feedback loop for modeling conventions and semantic annotation practices.

7.4. Limitations and Threats to Validity

Several limitations should be acknowledged.

Dependence on semantic alignment and model quality: Automated validation presupposes that the modeled elements relevant to a principle are represented and that the vocabulary is aligned to ontology concepts. Incomplete models, inconsistent naming, or missing annotations can lead to non-detections or results that reflect missing modeling information rather than real-world non-compliance.

Expressiveness and operationalization of principles: Not all EA principles are equally amenable to executable constraints. Principles that rely on qualitative judgement, context-specific trade-offs, or exception policies require additional operational definitions and may remain partially manual. The evaluated principles are predominantly governance and data/application consistency rules that can be expressed as structural constraints.

Closed-world interpretation: SHACL validation is typically executed under a closed-world assumption, which may treat absent knowledge as a finding. This is often appropriate for governance (e.g., missing responsibility links), but it requires careful interpretation when communicating results to stakeholders.

Generalizability: The evaluation spans two organizational settings and one educational setting. While this provides evidence of feasibility across contexts and modeling styles, broader generalization requires additional industrial case studies, larger model repositories, and further principle classes.

7.5. Future Work

Future work can extend the approach along four complementary dimensions.

First, stronger tool support for semantic alignment and annotation is required. Semi-automated mapping of model labels and textual principle statements to ontology concepts should be further developed. This includes exploring the use of large language models to support term extraction and ontology mapping, and designing collaboration mechanisms between LLM-based suggestions and the ontology validation layer to reduce the manual effort required for semantic grounding.

Second, reusable libraries of principle patterns and SHACL/SPARQL constraint templates should be developed. Such libraries could capture recurring governance patterns (e.g., uniqueness, single-authoritative-source, stewardship assignment, separation-of-concerns) and provide reusable SHACL and SPARQL constraint templates for typical EA governance scenarios. This would support cross-organizational reuse and facilitate faster adoption of automated validation in practice.

Third, explanation and remediation support should be enhanced. Beyond reporting violations, validation tools could provide structured guidance for exception handling, remediation suggestions, and integration with governance workflows and architecture review processes.

Fourth, scalability and maintenance effort should be systematically evaluated on larger, versioned EA repositories. This includes performance assessments of derivation and validation pipelines, as well as empirical studies on long-term maintenance of ontology extensions and constraint libraries, as well as analysis of incremental validation mechanisms for continuously evolving architecture models.

Finally, recent advances in large language models (LLMs) suggest opportunities for hybrid architectures in which AI agents support the refinement of governing statements into structured representations through competency-question-driven interactions. Such agents could assist in vocabulary harmonization, ontology alignment, and draft rule generation from textual documents, while maintaining a clear separation between AI-supported structuring and ontology-based validation to preserve transparency and auditability.

8. Conclusions

This paper investigated how semantic technologies can support automated validation of enterprise architecture (EA) models in accordance with EA principles. The presented approach combines (i) ontology-grounded ArchiMate model representations, (ii) SBVR Structured English for reducing ambiguity and preserving traceability of principle statements, (iii) a derivation layer to materialize implicit architectural relations required for robust checking, and (iv) SHACL/SPARQL constraint execution that produces explainable, traceable validation reports.

Across three case studies, the approach enabled repeatable compliance checks and produced results that can be inspected and discussed in governance contexts because violations are linked to concrete model elements. The findings highlight that reliable validation often depends on semantic completion: derivation rules are needed to compensate for heterogeneous modeling styles and for relations that are implied by ArchiMate semantics but not explicitly modeled. Overall, the study indicates that a layered transformation from natural-language principles to executable constraints can shift substantial parts of compliance checking from manual interpretation to automated validation, while keeping architectural judgement and exception handling in the loop.

In enterprise practice, the proposed method can directly support continuous EA governance by enabling automated and repeatable principle checks during the iterative evolution of architecture models. This reduces the effort required for manual architecture reviews while improving the transparency, traceability, and consistency of governance decisions through explicit links between principles, model elements, and validation outcomes.

Future research should expand the catalog of supported principle patterns, strengthen automation support for semantic alignment and annotation, and assess scalability and maintenance effort on larger industrial EA repositories and evolving principle sets. In addition, hybrid approaches that combine ontology-based validation with AI-assisted, competency-question-driven knowledge refinement warrant investigation. While large language models may support vocabulary harmonization, semantic alignment, and draft rule generation from textual documents, formal enterprise ontologies and executable SHACL/SPARQL constraints remain essential as the authoritative reasoning and traceability layer for governance auditability.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data supporting the reported results are available from the author upon reasonable request. Some data are not publicly available due to organizational confidentiality restrictions.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Zachman, J. A framework for information systems architecture. *IBM Syst. J.* **1987**, *26*, 276–292.
2. Ahlemann, F.; Stettiner, E.; Messerschmidt, M.; Legner, C. (Eds.) *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*; Springer Science+Business Media: Berlin/Heidelberg, Germany, 2012.
3. The Open Group. *TOGAF 9—The Open Group Architecture Framework V.9*; The Open Group: San Francisco, CA, USA, 2009.
4. The Open Group. *ArchiMate 3.1 Specification*; The Open Group: San Francisco, CA, USA, 2009.
5. van Gils, B.; Proper, E. Enterprise Modelling in the Age of Digital Transformation. In *The Practice of Enterprise Modeling*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 257–273.
6. Haki, M.K.; Legner, C. Enterprise architecture principles in research and practice: Insights from an exploratory analysis. In Proceedings of the European Conference on Information Systems (ECIS 2013) Completed Research, Utrecht, The Netherlands, 5–8 June 2013.
7. Haki, M.K.; Legner, C. The Mechanics of Enterprise Architecture Principles. *J. Assoc. Inf. Syst.* **2021**, *22*, 1334–1375.
8. Proper, E.; Hoppenbrouwers, S.; Veldhuijzen van Zanten, G.E. Communication of Enterprise Architectures. In *Enterprise Architecture at Work: Modelling, Communication and Analysis*, 4th ed.; Lankhorst, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 59–73.
9. Ahlemann, F.; Legner, C.; Schafczuk, A. Introduction. In *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*; Ahlemann, F., Stettiner, E., Messerschmidt, M., Legner, C., Eds.; Springer Science+Business Media: Berlin/Heidelberg, Germany, 2012; pp. 1–35.
10. Bridgeland, D.; Zahavi, R. *Business Modeling: A Practical Guide to Realizing Business Value*; Morgan Kaufmann: San Francisco, CA, USA, 2009.
11. Lankhorst, M.; Iacob, M.E.; Jonkers, H. State of the Art. In *Enterprise Architecture at Work: Modelling, Communication and Analysis*; Lankhorst, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 11–40.
12. Greefhorst, D.; Proper, E. *Architecture Principles: The Cornerstones of Enterprise Architecture*; Springer: Berlin/Heidelberg, Germany, 2011. https://doi.org/10.1007/978-3-642-20279-7.
13. Karagiannis, D.; Woitsch, R. Knowledge engineering in business process management. In *Handbook on Business Process Management 2*; vom Brocke, J., Rosemann, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 623–648.
14. Hinkelmann, K.; Gerber, A.; Karagiannis, D.; Thoenssen, B.; van der Merwe, A.; Woitsch, R. A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. *Comput. Ind.* **2016**, *79*, 77–86.
15. Kappel, G.; Kapsammer, E.; Kargl, H.; Kramler, G.; Reiter, T.; Retschitzegger, W.; Schwinger, W.; Wimmer, M. Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages. In *Model Driven Engineering Languages and Systems, Proceedings of the 9th International Conference, MoDELS 2006*, LNCS 4199; Nierstrasz, O., Whittle, J., Harel, D., Reggio, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 528–542.
16. Fill, H.G.; Schremser, D.; Karagiannis, D. A Generic Approach for the Semantic Annotation of Conceptual Models Using a Service-Oriented Architecture. *Int. J. Knowl. Manag.* **2013**, *9*, 76–88.
17. Hinkelmann, K.; Laurenzi, E.; Martin, A.; Thönssen, B. Ontology-based metamodeling. In *Business Information Systems and Technology 4.0*; Dornberger, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 177–194.
18. Hinkelmann, K.; Laurenzi, E.; Martin, A.; Montecchiari, D.; Spahic, M.; Thönssen, B. ArchiMEO: A Standardized Enterprise Ontology based on the ArchiMate Conceptual Model. In Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Valletta, Malta, 25–27 February 2020; INSTICC; SciTePress: Setúbal, Portugal, 2020; Volume 1, pp. 417–424. https://doi.org/10.5220/0009000204170424.
19. Borgers, M.; Harmsen, F. Strengthen the Architecture Principle Definition and Its Characteristics-A Survey Encompassing 27 Years of Architecture Principle Literature. In Proceedings of the ICEIS (2), Madeira, Portugal, 21–24 March 2018; pp. 607–622.
20. Marosin, D.; van Zee, M.; Ghanavati, S. Formalizing and Modeling Enterprise Architecture (EA) Principles with Goal-Oriented Requirements Language (GRL). In *Proceedings of the Advanced Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 205–220.
21. OMG. *Semantics of Business Vocabulary and Business Rules*, Version 1.5; Technical Report; Object Management Group: Milford, MA, USA, 2019.
22. Ross, R.G. *Business Knowledge Blueprints: Enable Your Data to Speak the Language of the Business (Issue 2)*; Business Rule Solutions LLC: Houston, TX, USA, 2020.

23. W3C RDF Schema Working Group. RDF 1.1 Primer. Website. 2014. Available online: https://www.w3.org/TR/rdf11-primer/ (accessed on 23 April 2019).
24. McGuinness, D.L.; Van Harmelen, F. OWL web ontology language overview. *W3C Recomm.* **2004**, *10*, 2004.
25. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member Submission* **2004**, 1–31.
26. Knublauch, H.; Kontokostas, D. *Shapes Constraint Language (SHACL)*; Technical Report; W3C: Shanghai, China, 2017.
27. Montecchiari, D. Ontology-based Validation of Enterprise Architecture Principles in Enterprise Models. In Proceedings of the BIR 2021 Workshops and Doctoral Consortium, Co-Located with 20th International Conference on Perspectives in Business Informatics Research, Vienna, Austria, 22–24 September 2021. Available online: http://ceur-ws.org/Vol-2991/ (accessed on 15 March 2026).
28. Montecchiari, D.; Hinkelmann, K. Towards Ontology-Based Validation of EA Principles. In *The Practice of Enterprise Modeling (PoEM)*; Springer International Publishing: Berlin/Heidelberg, Germany, 2022; pp. 66–81. https://doi.org/10.1007/978-3-031-21488-2_5.
29. Montecchiari, D.; Hinkelmann, K. Validating Enterprise Architecture Principles Using Derivation Rules and Domain Knowledge. In *Perspectives in Business Informatics Research*; Springer Nature: Cham, Switzerland, 2023; pp. 244–259.
30. Radeke, F.; Legner, C. Embedding EAM into Strategic Planning. In *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*; Ahlemann, F., Stettiner, E., Messerschmidt, M., Legner, C., Eds.; Springer Science+Business Media: Berlin/Heidelberg, Germany, 2012; pp. 110–139.
31. Ahlemann, F.; El Arbi, F. An EAM Navigator. In *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*; Ahlemann, F., Stettiner, E., Messerschmidt, M., Legner, C., Eds.; Springer Science+Business Media: Berlin/Heidelberg, Germany, 2012; pp. 35–55.
32. Aier, S.; Fischer, C.; Winter, R. Construction and Evaluation of a Meta-Model for Enterprise Architecture Design Principles. In Proceedings of the 10th International Conference on Wirtschaftsinformatik (WI 2011), Zurich, Switzerland, 16–18 February 2011; Bernstein, A., Schwabe, G., Eds.; Universität Zürich: Zurich, Switzerland, 2011; Volume 2.
33. Greefhorst, D.; Proper, E.; Plataniotis, G. The Dutch State of the practice of architecture principles. *J. Enterp. Archit.* **2013**, *9*, 20–25.
34. Greefhorst, D.; Proper, E. A practical approach to the formulation and use of architecture principles. In Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops, Helsinki, Finland, 29 August–2 September 2011; pp. 330–339.
35. Sandkuhl, K.; Simon, D.; Wißotzki, M.; Starke, C. The nature and a process for development of enterprise architecture principles. In *Business Information Systems*; Lecture Notes in Business Information Processing; Springer International Publishing: Cham, Switzerland, 2015; pp. 260–272.
36. Allemang, D.; Hendler, J. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, 2nd ed.; Morgan Kaufmann Publishers, Elsevier: San Francisco, CA, USA, 2011.
37. Ghani, I.; Lee, C.Y.; Juhn, S.H.; Jeong, S.R. Semantics-oriented approach for information interoperability and governance: Towards user-centric enterprise architecture management. *J. Zhejiang Univ. Sci. C* **2010**, *11*, 227–240.
38. Karagiannis, D.; Kühn, H. Metamodelling platforms. In *Proceedings of the EC-Web*; Citeseer: Princeton, NJ, USA, 2002; Volume 2455, p. 182.
39. Guarino, N. Formal Ontology and Information Systems. In *Proceedings of the Formal Ontology in Information Systems*; Proceedings of FOIS'98; Guarino, N., Ed.; Springer: Trento, Italy, 1998; pp. 3–15.
40. Antoniou, G.; van Harmelen, F. *A Semantic Web Primer*; MIT Press: London, UK, 2004.
41. Fox, M.S.; Barbuceanu, M.; Gruninger, M. An organisation ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour. *Comput. Ind.* **1996**, *29*, 123–134.
42. Uschold, M.; King, M.; Moralee, S.; Zorgios, Y. *The Enterprise Ontology the Knowledge Engineering Review*; Special Issue on Putting Ontologies to Use; Cambridge University Press: Cambridge, UK, 1998; Volume 13.
43. Bertolazzi, P.; Krusich, C.; Missikoff, M. An approach to the definition of a core enterprise ontology: CEO. In Proceedings of the OES-SEO 2001, International Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations, Rome, Italy, 14–15 September 2001; pp. 14–15.
44. Leppänen, M. A Context-Based Enterprise Ontology. In *Proceedings of the EDOC International Workshop on Vocabularies, Ontologies and Rules for the Enterprise (VORTE'05)*; Guizzardi, G., Wagner, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 17–24.
45. Kang, D.; Lee, J.; Choi, S.; Kim, K. An ontology-based enterprise architecture. *Expert Syst. Appl.* **2010**, *37*, 1456–1464.
46. Peter, M.; Montecchiari, D.; Hinkelmann, K.; Grivas, S.G. Ontology-Based Visualization for Business Model Design. In *Proceedings of the IFIP Working Conference on The Practice of Enterprise Modeling*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 244–258.
47. Laurenzi, E.; Hinkelmann, K.; Montecchiari, D.; Goel, M. Agile visualization in design thinking. In *New Trends in Business Information Systems and Technology*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 31–47.

48. Azzini, A.; Braghin, C.; Damiani, E.; Zavatarelli, F. Using Semantic Lifting for improving Process Mining: A Data Loss Prevention System case study. In Proceedings of the 3rd International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2013), Riva del Garda, Italy, 30 August 2013; pp. 62–73.
49. Hrgovcic, V.; Karagiannis, D.; Woitsch, R. Conceptual Modeling of the Organisational Aspects for Distributed Applications: The Semantic Lifting Approach. In Proceedings of the COMPSACW 2013, 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops, Kyoto, Japan, 22–26 July 2013; pp. 145–150. https://doi.org/10.1109/COMPSACW.2013.17.
50. Hinkelmann, K.; Laurenzi, E.; Lammel, B.; Kurjakovic, S.; Woitsch, R. A Semantically-Enhanced Modelling Environment for Business Process as a Service. In Proceedings of the 2016 4th International Conference on Enterprise Systems (ES), Melbourne, VIC, Australia, 2–3 November 2016; pp. 143–152. https://doi.org/10.1109/ES.2016.25.
51. OMG. *Meta Object Facility (MOF) Core Specification*, version 2.4.2; Technical Report; Object Management Group: Milford, MA, USA, 2014.
52. Laurenzi, E. An agile and ontology-based meta-modelling approach for the design and maintenance of enterprise knowledge graph schemas. *Enterp. Model. Inf. Syst. Archit.* **2024**, *19*, 6. https://doi.org/10.18417/emisa.19.6.
53. Schoonderbeek, J.A.H.; Proper, E. Toward an ontology for EA modeling and EA model quality. *Softw. Syst. Model.* **2024**, *23*, 535–558.
54. Corradini, F.; Polini, A.; Re, B.; Tiezzi, F. An Operational Semantics of BPMN Collaboration. In *Formal Aspects of Component Software*; Braga, C., Ölveczky, P.C., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2016; Volume 9539, pp. 161–180.
55. Ross, R.G. Basic RuleSpeak Do's and Don'ts, Version 2.2.5. Available online: https://www.rulespeak.com/en/Basic%20RuleSpeak%20Dos%20and%20Donts%20v2-2-5.pdf (accessed on 15 March 2026).
56. Pérez, J.; Arenas, M.; Gutierrez, C. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.* **2009**, *34*, 1–45.
57. Karpovic, J.; Nemuraite, L. Transforming SBVR business semantics into Web ontology language OWL2: Main concepts. *Inf. Technol.* **2011**, 27–29.
58. Reynares, E.; Caliusco, M.L.; Galli, M.R. SBVR to OWL 2 mappings: An automatable and structural-rooted approach. *CLEI Electron. J.* **2014**, *17*, 2:1–2:18.
59. Reynares, E.; Caliusco, M.L.; Galli, M.R. A set of ontology design patterns for reengineering SBVR statements into OWL/SWRL ontologies. *Expert Syst. Appl.* **2015**, *42*, 2680–2690.
60. Pham, T.A.; Thanh, N.L. An ontology-based approach for business process compliance checking. In Proceedings of the Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication, Danang, Viet Nam, 4–6 January 2016; pp. 1–6.
61. Hevner, A. A three cycle view of design science research. *Scand. J. Inf. Syst.* **2007**, *19*, 4.
62. Peffers, K.; Tuunanen, T.; Rothenberger, M.A.; Chatterjee, S. A Design Science Research Methodology for Information Systems Research. *J. Manag. Inf. Syst.* **2008**, *24*, 45–77.
63. Drechsler, A.; Hevner, A. A Four-Cycle Model of IS Design Science Research: Capturing the Dynamic Nature of IS Artifact Design. In Proceedings of the Breakthroughs and Emerging Insights from Ongoing Design Science Projects: Research-in-Progress Papers and Poster Presentations from the 11th International Conference on Design Science Research in Information Systems and Technology (DESRIST), St. John's, NL, Canada, 23–25 May 2016; pp. 1–8.
64. Goldkuhl, G. Design Research in Search for a Paradigm: Pragmatism is the Answer. In *Proceedings of the Practical Aspects of Design Science*; Helfert, M., Donnellan, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 84–95. https://doi.org/10.1007/978-3-642-33681-2_8.
65. Neuman, W.L. *Social Research Methods: Qualitative and Quantitative Approaches*; Pearson: London, UK, 2014.
66. Laurenzi, E. AOAME: An Enterprise Knowledge Graphs Editor for Domain Experts. In *Intelligent Information Systems*; Springer: Cham, Switzerland, 2025; pp. 181–188. https://doi.org/10.1007/978-3-031-94590-8_22.