EPH- International Journal of Science And Engineering
ISSN (Online): 2454-2016
Volume 12 Issue 01 January 2026

DOI: https://doi.org/10.53555/wfehqv74

INTEGRATING GRAPH DATABASE MODELING INTO AGILE SYSTEM ANALYSIS AND DESIGN: A CONCEPTUAL FRAMEWORK FOR SCALABLE INFORMATION SYSTEMS

Brenda M. Balala[1,2], Dr. Reagan Ricafort[3]

[1]AMA Education System Quezon City, Philippines Brenda.Balala@live.ama.edu.ph
[2]Notre Dame of Marbel University, Koronadal City, Philippines bmbalala@ndmu.edu.ph
[3]Dean, Graduate School, AMA Education System

Corresponding Author:
bmbalala@ndmu.edu.ph

ABSTRACT
Relational database modeling is still the most popular in system analysis and design (SAD), but has structural constraints in relationship-intensive domains like social networks, fraud detection, recommendation systems, and knowledge graphs (Angles & Gutierrez, 2008; Robinson et al., 2015) . These constraints are aggravated in agile systems with iterative delivery and emergent design (Beck et al., 2001; Fowler, 2002; Schwaber & Sutherland, 2017) , in which architectural decisions that are susceptible to traversal are often deferred.
This study presents the Graph-Agile Framework, a design science artifact that brings the concepts of graph database modeling into agile SAD using a three-tier architecture: Strategic Alignment, Tactical Integration, and Operational Adaptation. Grounded in design science research and theory synthesis (Gregor & Hevner, 2013; Hevner et al., 2004; Jaakkola, 2020a) , the framework institutionalizes traversal-aware reasoning, schema optionality, and sprint-level scalability validation into the iterative workflows.
An expert-based evaluation (N = 40) assessed structural clarity, theoretical grounding, agile alignment, scalability support, and adoption feasibility. Findings indicate consistently high expert-rated internal coherence and compatibility with agile practices, providing preliminary perceptual support for the framework's utility.
The study advances SAD theory beyond relational assumptions and extends agile methodology by formalizing traversal-aware architectural governance for scalable graph-centric system development.

Keywords: *graph database modeling, agile system analysis and design, property graph model, schema evolution, scalable information systems, iterative development*

1. INTRODUCTION

1.1 Background and Rationale

In modern information systems, value increasingly derives from relationships between entities rather than from isolated data objects. Social networking, fraud detection, recommender systems, knowledge management, and biometrics are only a few areas where complex network analysis is essential to identify structural patterns and emergent information (Aalabaf-Sabaghi, 2012; Newman, 2018; Rajaraman & Ullman, 2011; Fensel et al., 2020); Pavlopoulos et al., 2011). In such contexts, efficient traversal and relationship-centric querying are essential for scalability and analytical effectiveness.

Relational database management systems (RDBMS), the basis of traditional system analysis and design (SAD), were initially developed to complement attribute-focused data and normalized structures that are optimized to handle transactional applications (Codd, 1970;Date, 2003; Elmasri, Ramez and Navathe, 2016). Relational systems despite their high performance and scalability are highly expensive in regard processing that requires relationship-intensive queries particularly at high level so joins despite their capability to process structured data (Angles & Gutierrez, 2008; Bonifati et al., 2018).

Graph databases are specifically designed to handle highly connected domains by their flexible, index-free adjacency, and rich semantics, as well as their native graph algorithms, which scale with size of the result set instead of the size of the underlying data (Robinson et al., 2015; Angles & Gutierrez, 2008).

At the same time, agile approaches have been adopted as the new paradigm of developing software and information systems with a strong focus on iteration, cooperation with stakeholders, and emergent design (Beck et al., 2001; Cockburn, 2006; Nathan & Scobell, 2017). However, graph database performance depends on early architectural decisions, such as traversal patterns, relationship directionality, and data locality, which are expensive to update after deployment (Robinson et al., 2015; De Virgilio et al., 2014). The available agile SAD practice does not offer much information on how such graph-related issues ought to be tackled during requirements analysis and design.

The following difficulties inspire the creation of a well-organized Graph-Agile Framework conceptual model, outlined in Sections 3 and 4, that incorporates the principles of graph database modeling in the agile system analysis and design.

1.2 Problem Statement

These are common challenges that organizations face in agile settings when implementing graph databases:

1. Inadequate Design Artifacts

Traditional Entity-Relationship Diagrams (ERDs) and normalized schemas are structurally grounded in relational presumptions and do not resistively depict traversal semantics, relationship properties, and recursive or multi-hop paths (Chen, 1976; Angles & Gutierrez, 2008; Bonifati et al., 2018).

2. Design Philosophy Mismatch

The agile emergent design principles place focus on incremental architectural refinements in architecture that could conflict with the requirement of early traversal-aware architectural choices in graph systems, in which the directionality of relationships and locality to paths play a critical role in scalability (Beck, 1999; Fowler, 2004; Robinson et al., 2015).

3. Skills Gap

Analysts and developers who are mostly trained primarily in relational modeling concepts lack proficiency in graph-native abstractions, traversal-oriented reasoning, and property graph query languages (Angles, 2012; Francis et al., 2018).

4. Scalability Validation Constraints

Graph-centric benchmarking, traversal complexity testing, and path-performance analysis are not commonly used as conventional agile validation techniques throughout sprint cycles (Vicknair et al., 2010; Iosup et al., 2015).

5. Tooling Limitations

Available agile tools and documentation practices are not very helpful in supporting graph-native modeling, traversal visualization, and schema evolution governance (Holzschuher & Peinl, 2013).

1.3 Research Objectives

This study aims to:

1. Formulate a conceptual model combining agile SAD and graph database modeling.
2. Synthesize a theory of graph database and agile principles.
3. Identify required changes on agile artifacts and ceremonies for graph-centric systems.
4. Provide guidance for practitioners, researchers, and educators.

1.4 Research Questions

1. How can graph database modeling principles be reconciled with agile system analysis and design?
2. How should agile artifacts and ceremonies be adapted to support graph-centric development?
3. What is the effect of graph-integrated agile SAD on scalability and the complexity of design?
4. Which organizational attributes influence the adoption of the Graph-Agile Framework?

1.5 Significance of the Study

The research paper adds to the information systems theory by making the graph database modeling a specific design paradigm in system analysis and design, a formalized one (Wand & Weber, 2002; Burton-Jones et al., 2017). In approach, it has expanded agile methods to support relationship-based methods. Methodologically, it extends agile practices to accommodate relationship-centric data structures (Fowler, 2004; P. Abrahamson et al., 2002). Practically, it provides analysts and development teams with actionable guidance for adopting graph databases without abandoning agile principles. The research can also be used in the development of a curriculum by defining competencies needed in the Graph-Agile Framework.

2. LITERATURE REVIEW

2.1 Graph Database Modeling Paradigms

Graph databases use the property graph model, where data are defined as nodes, relationships, and properties. In contrast to relational models, relationships are treated as path-based first-class constructs, which makes adjacency index-free and allows traversing multiple hops efficiently (Angles, 2012; Robinson et al., 2015). Due to this, the traversal complexity is more reliant on the result set size and not the underlying dataset size (Angles & Gutierrez, 2008).

Empirical and analytical studies indicate that property graph models are more effective in relationship-intensive applications and offer better modeling expressiveness to domains with a complex pattern of connectivity (Bonifati et al., 2018 ; Robinson et al., 2015) . All this makes graph databases particularly suitable into the applications where recursive, variable-length and path based queries are expected to be performed.

2.2 Limitations of Relational Modeling

Relational systems have join explosion, schema rigidity, and restricted ability to perform recursive and variable-length path queries. With a deeper traversal, query complexity rises in a combinatorial fashion, limiting scalability and analytical capability in large-scale networks. Recursive and non-fixed queries also contribute to the complexity of joins and the worsening of performance (Bonifati et al., 2018; Vicknair et al., 2010).

2.3 Agile System Analysis and Design

The agile methodologies are concerned with incremental delivery, collaboration with stakeholders, and emergent design through the use of artifacts such as user stories, product backlogs, sprint planning, and retrospectives (Beck et al., 2001; Cockburn, 2006). Although the practices are effective in dealing with the changing requirements, traditional SAD artifacts (mainly UML diagrams and ERD) are those of relational data models and lack the power to express graph-centric systems (Fowler, 2004;Ambler, 2003).

Consequently, agile SAD presents minimal advice on modeling traversal semantics, relationship properties, and schema change in graph databases, supporting the necessity of modified artifacts of analysis and design practices.

2.4 Conceptual Frameworks in Information System Design

Design science studies offer an empirical method of creating prescriptive models that combine theory and practice. Integration of findings across fragmented areas of research is enabled by theory synthesis, which helps generate explanatory and action-oriented conceptual models. Design science research enables prescriptive artifact development (Hevner et al., 2004) , Theory synthesis facilitates integration of fragmented literature streams into coherent Information Systems Success (ISS) frameworks (Gregor, 2006; Jaakkola, 2020).

2.5 Synthesis and Research Gap

Conversely, the agile system analysis and design literature emphasizes emergent design, lightweight artifacts, and iterative refinement (Beck et al., 2001; Cockburn, 2006; Fowler, 2004). Nevertheless, these practices are mostly based on relational assumptions with reliance on UML-based and ERD-centric representations that

inadequately capture traversal semantics, relationship properties, and schema optionality inherent in graph-based systems (Ambler, 2003; Fowler, 2004).

Despite the recent research recognizing the emerging use of graph databases in agile contexts (Angles, 2012; Dominguez-Sal et al., 2010; Dong et al., 2024), graph modeling remains largely absent from formal SAD methodologies. As a result, existing approaches provide little guidance on how agile artifacts, ceremonies, and validation practices should be adapted to support traversal-oriented modeling, early scalability reasoning, and continuous schema evolution (Robinson et al., 2015; Iosup et al., 2015).

3. METHODOLOGY

3.1 Research Design

The paper employed a Design Science Research (DSR) paradigm to develop and test a prescriptive conceptual framework model with the use of graph database modeling in agile system analysis and design (Gregor & Hevner, 2013; Hevner et al., 2004). The study is placed in the rigor relevance cycle of DSR that focuses on the utility of the artifact, theoretical background, and methodological consistency.

In graph database modeling and agile techniques, theory synthesis has been used to bring together diasporic bodies of work in research into a single conceptual artifact (Jaakkola, 2020a). The study adheres to the accepted guidelines of conceptual framework development, such as identification of central constructs, definition of relationships as well as formulation of theoretically based propositions (Whetten, 1989). The resulting artifact is evaluated for conceptual clarity, internal coherence, and perceived practical applicability rather than causal performance outcomes.

3.2 Literature Search and Selection

The literature was selected based on its relevance to the research objectives through a systematic analysis of pertinent academic sources. Peer-reviewed journal articles, conference proceedings, and foundational texts on the topic of AIS were systematically searched in major academic databases, such as Scopus, Web of Science, IEEE Xplore, ACM Digital Library, and AIS Electronic Library. The search terms were based on combinations of the keywords: graph databases, property graph model, agile system analysis and design, schema evolution, agile modeling, and scalable information systems.

Inclusion criteria required relevance to:

- Graph database conceptualization and performance characteristics
- Agile strategies and SAD artifacts
- Refactoring, iterative design, and schema evolution
- Information systems research conceptual or methodological frameworks

This corpus was used to lay the theoretical framework of the synthesis of the design principles of graph-native with agile SAD practices.

3.3 Framework Development Process

Figure 1: Framework Development Process Using Design Science Research and Theory Synthesis

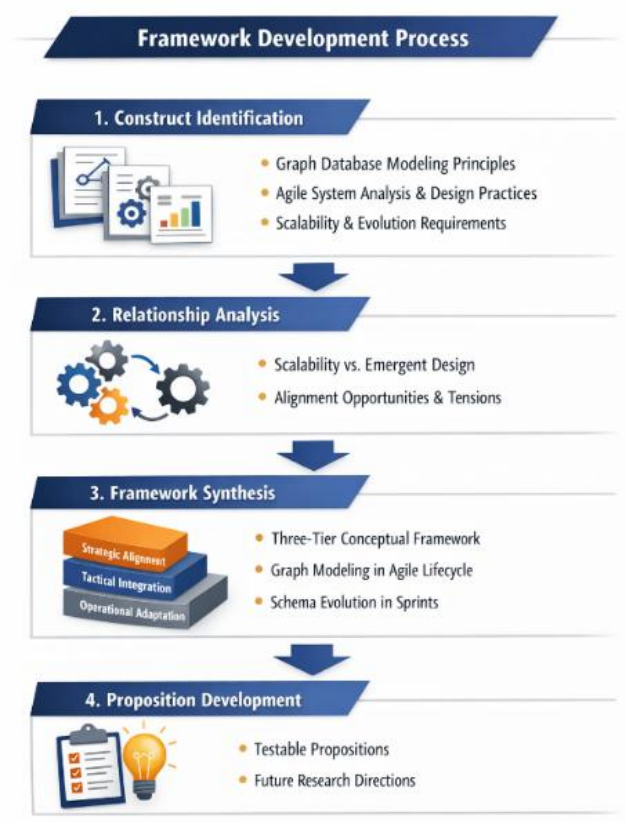


Figure 1 Framework Development Process (Methodology)

As the DSR methodology (Gregor & Hevner, 2013; Hevner et al., 2004), implies, the framework was created in a four-stage process.

Stage 1: Construct Identification
Core constructs were derived from the literature and grouped into three conceptual domains:
1. **Graph Database Modeling Principles** – including property graph models, traversal patterns, relationship semantics, schema optionality, and traversal optimization (Angles & Gutierrez, 2008; Robinson et al., 2015).
2. **Agile System Analysis and Design Practices** – user stories, sprint planning, incremental delivery, and continuous feedback (Beck et al., 2001; Fowler, 2004).
3. **Scalability and Evolution Requirements** – such as incremental validation, benchmarking, and structural refactoring (Iosup et al., 2015) .

Stage 2: Relationship Analysis
The analytical research to establish the tensions and complementarities between the principles of emergent design agility and graph architecture is required. The focus was made on the area of traversal planning, relationship directionality, and schema evolution, in which delayed architectural reasoning could be costly to performance (Robinson et al., 2015).

Stage 3: Framework Synthesis
The constructs and their interrelationships were synthesized into an integrated conceptual framework:
- **Strategic Alignment** – mapping business aims to traversal conscious requirements (Needham & Hodler, 2020).
- **Tactical Integration** – embedding property graph representations and traversal-oriented denormalization within agile artifacts (S. Ambler & Sadalage, 2006; Robinson et al., 2015).
- **Operational Adaptation** – introducing the operational adoption of schema evolution at the sprint-level schema evolution, traversal benchmarking, and query optimization (Francis et al., 2018; Holzschuher & Peinl, 2013).

This synthesis reflects the iterative search process central to DSR (Hevner et al., 2004).

3.4 Graph-Specific Propositions

Based on the synthesized conceptual framework, the following theoretically informed propositions have been put forward in order to be empirically validated:

P1: Evidence-based prototypes in the form of graphs in sprint cycles enhance scalability validation as they make traversal benchmarking during the iterative development process (Iosup et al., 2015).

P2: Schema-optional graph models are based on the principles of agile emergent design, which makes it easier to evolve the structure incrementally without strongly specified specifications (Robinson et al., 2015).

P3: Early traversal-based modeling at requirements engineering reduces post-deployment refactoring of connectivity-intensive systems.

These propositions are expected relationships among traversal conscious modeling practices and agile development achievements. Although the study in question gives a perceptual evaluation of these relationships, behavioral and performance confirmation is an aspect of future research.

3.5 Evaluation Design

3.5.1 Evaluation Strategy

In line with the conceptual artifact evaluation practices in the information systems research (Gregor & Hevner, 2013; Hevner et al., 2004) , the framework was evaluated in terms of perceived utility, internal consistency, and methodology fit with expert evaluation in a structured form.

Seven theoretically-based dimensions (1 = strongly disagree; 5 = strongly agree) were measured using a 5-point Likert scale instrument:

- Structural clarity
- Theoretical grounding
- Methodological alignment with agile SAD
- Practical applicability
- Scalability support
- Agile compatibility
- Adoption feasibility

The derivation of dimensions is in line with the principles of conceptual evaluation that are described in Jaakkola, 2020a) and Whetten, 1989).

3.5.2 Respondent Profile

It involved forty experts (N = 40) comprising of academic researchers, system analysts, programmers, and industry practitioners. Despite the fact that 72.5% has restricted direct graph database implementation experience, participants had pertinent experience in agile and system design settings to help them make informed judgment about methodological coherence.

3.5.3 Statistical Analysis and Reliability

In order to increase the methodological rigor, internal consistency reliability was calculated with **Cronbach's alpha** for each multi-item dimension. Cronbach's alpha (α) is a measure of internal consistency of multi-item constructs because it estimates the extent to which items measure a common underlying dimension. Constructs were all above the recommended 0.70 threshold, indicating acceptable reliability of the constructs to be used in the exploration of perceptual value.

There were some descriptive statistics (mean and standard deviation) calculated on each dimension. The coefficient of variation was analyzed to evaluate the dispersion in responses and showed rather low variability and a uniform agreement between dimensions.

The Pearson correlation was used to study the relationships between the variables of scalability support and the agile methodological alignment, with positive relationships found as per the theory (Iosup et al., 2015; Robinson et al., 2015).

Since the artifact was conceptual and exploratory, the test of the inferential hypothesis was not used. The results should be viewed as the preliminary perceptual validation in accordance with the standards of DSR evaluation (Gregor & Hevner, 2013).

3.5.4 Validity Considerations

Construct Validity:

Dimensions of measurement were defined based on theoretically defined constructs, and this guarantees the correspondence between the instrument of evaluation and the conceptual model (Jaakkola, 2020a; Whetten, 1989).

Internal Validity:
Quantitative ratings and qualitative feedback are more useful in the process of triangulation which increases interpretive credibility.
External Validity:
There are perceptual evaluations and asymmetry of expertise that limit the generalizability. The future empirical case studies and comparative performance analysis is required (Iosup et al., 2015; Robinson et al., 2015).
3.6 Methodological Limitations
The main weakness of the research is that it is based on the assessment of perceptual experts instead of being validated by longitudinal behavioral validation. Though the conceptual rigor and coherence are achieved through the use of experts, the validity in real-world agile graph implementation is to be achieved through empirical validation.
Future studies should incorporate:

- Action research and industrial case studies
- Relative comparison of relational-agile and Graph-Agile Framework
- Development of Graph-Agile Framework
- Domain-specific validation (e.g., knowledge graphs, fraud detection systems)

Stratified sampling in future studies may also address expertise asymmetry within respondent populations.
To demonstrate methodological alignment and rigor, Table 3 maps the proposed Graph-Agile Framework against the design science research guidelines articulated by Hevner et al. (2004).
Table 3 summarizes the correspondence of the Graph-Agile Framework with Hevner's Design Science Research (DSR) guidelines.

Table 3. Correlation of the Graph-Agile Framework with Hevner's Design Science Research Guidelines

DSR Guideline (Hevner et al., 2004)	Description	Alignment in This Study
1. Design as an Artifact	DSR must produce a viable artifact (construct, model, method, or framework).	The study constructs the Graph-Agile Framework as a conceptual artifact prescriptive combination of graph database modeling with agile SAD using articulated constructs (traversal awareness, schema optionality), relationships (tier integration), and propositions (P1-P3).
2. Problem Relevance	The artifact must address a relevant, unsolved organizational problem.	The framework solves structural constraints in the relational SAD in systems that are connectivity-centered, and the methodological gap in agile direction of graph-native architectures.
3. Design Evaluation	The artifact must be rigorously evaluated.	The study assesses the structure by administering a structured survey among experts on the structural clarity, theoretical grounding, agile alignment, scalability support, and adoption feasibility. The results have offered perceptual confirmation of the coherence and usefulness of the artifacts, while acknowledging that behavioral confirmation is required in the future.
4. Research Contributions	The research must provide clear and verifiable contributions.	The study provides contributions as (1) extending SAD beyond relational assumptions, (2) integrating graph modeling into agile workflows, and (3) proposing a structured governance model for traversal-aware development.
5. Research Rigor	The artifact must be grounded in rigorous theory and methodology.	The framework will be founded on the graph database theory, agile SAD literature, and theory synthesis within a DSR paradigm. Construct articulation and proposition development are based on set conceptual modeling standards.

DSR Guideline (Hevner et al., 2004)	Description	Alignment in This Study
6. Design as a Search Process	The artifact emerges through iterative refinement.	The framework was the result of the synthesis of tensions between emergent agile design and graph architecture constraints, and was worked out by theoretical concatenation and feedback by experts.
7. Communication of Research	Design science research must be communicated effectively to both technology-oriented and management-oriented audience, demonstrating the artifact's utility, rigor, and relevance in forms to each community.	The study presents the framework containing the three-tier architecture that is well structured, articulated propositions, and implications that apply to the practice and are accessible to both research, educational and industry domains.

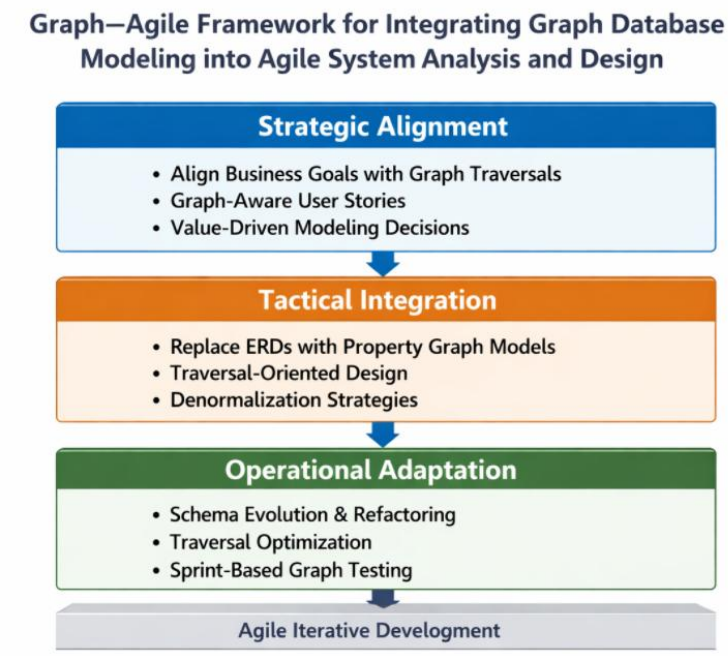
4. RESULTS: THE PROPOSED FRAMEWORK

4.1 Framework Overview

Graph-Agile Framework is the application of Agile system analysis and design using the principles of graph database modeling and formal three-layered architecture: Strategic Alignment, Tactical Integration, Operational Adaptation. Each of these layers interacts and integrates to introduce traversal-based reasoning across the agile lifecycle without violating the principles of iterative development and emergent design (Gregor & Hevner, 2013; Needham & Hodler, 2020).

The framework builds on agile SAD instead of proposing a parallel approach to the methodology, by taking into account graph-native architectural concerns at strategic, design, and operational aspects. Figure 2 illustrates how traversal-aware requirements, graph-native artifacts, and continuous schema evolution interact recursively within sprint cycles. The expert-based assessment of the structural clarity, theoretical underpinning, and practical applicability of the framework is provided in Section 4.7.

Figure 2:



4.2 Strategic Alignment

Strategic Alignment provides the normative basis of the Graph-Agile Framework as it will assure that adoption of graph databases will be carried out owing to organizational value and not due to technological experimentation. On this level, strategic goals are translated into traversal-conscious requirements, which consider that graph databases gain analytical capabilities as a result of modeling structural connectivity and multi-hop relationships (Newman, 2018; Robinson et al., 2015).

At this layer, traversal semantics are treated as an element of abstraction that informs strategic architectural decision-making. The business questions are also rephrased as connectivity issues, the directedness of relationships, cardinality expectations, and performance constraints are well-defined in user stories (Cohn, 2004; Needham & Hodler, 2020) . By integrating the matters of traversal into acceptance requirements, architectural foresight is established without compromising agile ability.
The framework is not substituting the agile methodologies, but instead, it builds agile SAD by integrating the graph-specific architectural rationale into the existing iterative practices (Beck et al., 2001; Dingsøyr et al., 2012) . Through institutionalizing the continual reevaluation of traversal pattern and relationship semantics, the framework balances the emergent design with the architectural foresight (Dong et al., 2024; Fowler, 2004).
Thus, the governance system that would provide the balance between the flexibility and structural intentionality in graph-based systems would be iterative development.

4.3 Tactical Integration

Tactical Integration graphically implements graph-native abstraction within SAD artifacts. It raises the Property Graph Model to the main conceptual representation, where the nodes, relationships, and properties as ontologically equal constructs (Angles, 2012; Angles & Gutierrez, 2008).
Although relational design focuses on eliminating redundancy by normalizing tables, this design builds upon traversal-oriented denormalization as a design heuristic. The structure of schemas is determined by expected patterns of traversal and query requirements instead of abstract entity taxonomies (Bonifati et al., 2018; Robinson et al., 2015) . This change is a paradigmatic change to connectivity-centric abstraction as opposed to entity-centric modeling.
Based on agile modeling and evolutionary database design (S. Ambler & Sadalage, 2006; S. W. Ambler, 2003) , Tactical Integration introduces graph-specific constructs into the working cycles. Traditional ERDs can also be supplemented or replaced by property graph diagrams that explicitly represent path dependencies and relationship semantics. This ensures architectural coherence while preserving agile flexibility.
Theoretically, this layer restructures the normalization theory based on a graph-native view and makes query based schema design as a formal modeling principle.

4.4 Operational Adaptation

Operational Adaptation embeds graph-specific optimization and schema evolution within iterative sprint cycles. Considering that the graph schemas are schema-optional and can be extended incrementally, this layer makes traversal benchmarking, query optimization, structural refactoring, and so on a regular sprint-level practice (Holzschuher & Peinl, 2013; Robinson et al., 2015).
The optimization of traversal with the help of the Cypher or Gremlin language is viewed as an ongoing engineering activity, but not an active corrective practice (Francis et al., 2018) . Graph-specific testing is included in sprint reviews and is used to fix the validation shortcomings in traditional agile processes (Iosup et al., 2015).
In theory, this layer brings agile concepts of continuous improvement and technical excellence to the world of graph-native architectures (Fowler, 2004; Schwaber & Sutherland, 2017). This makes schema evolution a controlled structural iteration as opposed to ad hoc modification, which is designed to enhance scalability within graph-native environments.

4.5 Agile Iterative Development as the Foundation

Agile Iterative Development functions as the temporal and epistemological mechanism through which Strategic Alignment, Tactical Integration, and Operational Adaptation are recursively implemented. Each sprint cycle functions as a bounded experimentation environment in which traversal-aware requirements, graph-native artifacts, and schema refinements are incrementally validated and refined.
In this design, traversal reasoning is incorporated into iterative processes and is not handled as an architectural stage. Strategic Alignment is used to inform backlog refinement by converting business goals into traversal-aware user stories. The requirements are implemented in the forms of property graph representations and traversal oriented schema heuristics under the name of Tactical Integration. Operational Adaption makes continuous benchmarking, refactoring and query optimization a part of the sprint review and retrospective institutionally. Combining these layers forms a feedback loop, which maintains agility and makes architectural intentionality (Beck et al., 2001; Dingsøyr et al., 2012). By institutionalizing continuous reassessment of traversal patterns and relationship semantics, the framework reconciles emergent design with architectural foresight (Dong et al., 2024; Fowler, 2004).

Therefore, iterative development would be the governance system that balances the flexibility and structural intentionality in graph-based systems.

4.6 Theoretical and Practical Significance

Theoretical Contribution

The Graph-Agile Framework also adds to the system analysis and design (SAD) through the expansion of the current modeling perspectives towards supporting graph-native architectural requirements. Instead of looking at graph databases as a technical extension of relational systems, the framework places them within a structured agile integration framework that is anchored on design science concepts (Hevner et al., 2004; Jaakkola, 2020a).

On a theoretical scale, the framework forms new relations between graph modeling and agile processes, developing propositions and boundary conditions that should be followed in future empirical validation.

Practical Contribution

In practice, the framework leaves a systematic way of embracing the use of graph databases in an agile setting. It mitigates the risks of late-stage refactoring and bottlenecks in scalability since it incorporates the modeling-specific concerns on strategic, tactical, and operational levels (Iosup et al., 2015; Robinson et al., 2015).

The expert-based evaluation provides preliminary perceptual support for Propositions P1 and P2, indicating that sprint-level graph prototyping is associated with a better validation of scalability and that schema-optional modeling is in line with agile emergent design concepts.

4.7 Framework Evaluation

Graph-Agile Framework was reviewed in an attempt to determine the conceptual clarity, theoretical background, alignment between the methodological approach and agile SAD, applicability, and support of scalability. Within the rigor cycle of design science research, the evaluations focused on the utility, coherence, and relevance of the design-based research, but not on the causal hypothesis testing (Gregor & Hevner, 2013; Hevner et al., 2004).

A mixed-method approach that involved quantitative evaluation with structured assessment and expert feedback on a qualitative basis enhanced the interpretive validity with a triangulation approach.

4.7.1 Evaluation Method

The form of 5-point Likert scale (1 = strongly disagree; 5 = strongly agree) was used among 40 professionals in software engineering, system analysis, agile development, and graph database technologies. The tool reflected seven dimensions that were theorized:

- Structural clarity
- Theoretical grounding
- Methodological alignment with agile SAD
- Practical applicability
- Scalability support
- Agile compatibility
- Adoption feasibility

This method fits the current practices of evaluating the conceptual artifact in the context of information systems research, where expert opinion is employed to evaluate the explanatory power, internal consistency, and practical significance (Jaakkola, 2020a; Whetten, 1989).

4.7.2 Respondent Profile and Expertise

The sample of respondents was academic gurus (45 %) and system analysts (15 %), programmers/ entry-level practitioners (35 %t), and industry practitioners (5 percent). Although 72.5% responded to the question directly as they lacked the implementation of graph databases, the participants were best informed on the topic of implementation of agile and system design, and this aids in determining the compatibility of the research methodology.

It increases the credibility of the interpretation and also guarantees the relevance to domains since there are diverse roles and levels of experience. Interpretive credibility is enhanced by virtue of the diversity of roles and level of experience, whereas domain relevance is preserved.

Mean scores are above 4.30 across all of these dimensions, which suggests that there is a good level of consensus among experts on the area of conceptual clarity, methodological compatibility, and practical relevance.

The ratings of structural clarity and agile alignment were rated highly, and this indicates that the traversal-conscious reasoning becomes naturally incorporated into the processes of iterative development. The adoption feasibility is not as high, but it is also positively high, and could be the expected learning curves, but not probably structural limitations.

The findings provide preliminary perceptual support for the following propositions.

Notably, reviews illuminated contextualization as opposed to questioning the theoretical soundness. The architecture best fits connectivity-intensive systems and can add needless overhead to a totally transactional environment.

Table 1: Respondent Profile Summary (N=40)

Attribute	Category	Frequency	Percentage
Professional Role	Academic Expert	18	45.0%
	System Analyst	6	15.0%
	Industry Practitioner	2	5.0%
	Programmer/Entry-Level	14	35.0%
Years of Experience	1-3 years	25	62.5%
	4-7 years	8	20.0%
	8-12 years	1	2.5%
	>12 years	6	15.0%
Graph DB Experience	Yes	11	27.5%
	No	29	72.5%
Self-Assessed Expertise	Beginner	9	22.5%
	Intermediate	29	72.5%
	Advanced	2	5.0%

4.7.3 Quantitative Evaluation Results

Table 2: Descriptive Statistics by Evaluation Dimension (N = 40)

Dimension	No. of Items	Cronbach's α	Mean (M)	SD	Interpretation
Structural Clarity	4	0.866	4.57	0.58	Very High
Theoretical Grounding	4	0.849	4.53	0.63	Very High
Agile Methodological Alignment	5	0.857	4.61	0.61	Very High
Practical Applicability	5	0.880	4.56	0.62	Very High
Scalability & Traversal Awareness	4	0.833	4.60	0.60	Very High
Adoption Feasibility	3	0.763	4.39	0.77	Very High

Across all dimensions, mean scores exceeded 4.30, indicating strong expert agreement regarding conceptual clarity, methodological compatibility, and practical relevance.

Structural clarity and agile alignment received particularly strong ratings, suggesting that traversal-aware reasoning integrates cohesively within iterative development practices. Adoption feasibility, while comparatively lower, remains strongly positive and likely reflects anticipated learning curves rather than structural limitations.

These findings provide preliminary perceptual support for:

- **P1:** Sprint-level graph prototyping enhances scalability validation.
- **P2:** Schema-optional modeling aligns with agile emergent design.

4.7.4 Qualitative Feedback Synthesis

Qualitative feedback converged with quantitative results. Respondents emphasized:

- Coherence of the three-tier architecture
- Clear mapping of business objectives to traversal-aware requirements
- Compatibility with existing agile workflows

Recommendations focused on developing modeling templates, workflow exemplars, and phased adoption pathways for teams transitioning from relational paradigms.

Importantly, critiques clarified contextual boundaries rather than challenging theoretical validity. The framework is most appropriate for connectivity-intensive systems and may introduce unnecessary overhead in purely transactional environments.

5. DISCUSSION

5.1 Integration of Findings and Research Questions

In this study, it is demonstrated that graph database modelling is compatible with agile system analysis and design (SAD) by systematically integrating methodologies as opposed to methodologically replacing each other. The proposed Graph-Agile Framework integrates traversal-aware thinking in 3 levels of interdependence, i.e., Strategic Alignment, Tactical Integration, and Operational Adaptation, and, thus, overcomes architectural tension defined in Section 1.2, such as artifact inadequacy, philosophical mismatch between emergence and foresight, scalability validation gaps, and tooling limitations (Angles & Gutierrez, 2008; Iosup et al., 2015; Robinson et al., 2015).

Good scalability and agile compatibility ratings of experts indicate that scalability-aware modeling augments agile response, but does not limit it. This observation goes in line with the studies that propose that agile procedures should be sensitive to domain-specific architectural limitations in complex systems (Boehm, 2002; Fowler, 2004).

The research questions are addressed as follows:

- **RQ1:** Graph modeling and agile SAD can be reconciled by formalizing traversal semantics within user stories and acceptance criteria, thereby integrating connectivity reasoning into iterative workflows (Cohn, 2004; Needham & Hodler, 2020).
- **RQ2:** Traditional ERDs and normalization-centric artifacts may be supplemented or replaced with property graph representations and traversal-oriented denormalization heuristics (Angles, 2012; Robinson et al., 2015).
- **RQ3:** Sprint-level graph prototyping improves the validation of scalability through incorporating traversal benchmarking into the iterative cycles (Iosup et al., 2015).
- **RQ4:** Adoption feasibility is influenced primarily by organizational readiness and modeling expertise rather than structural incompatibility between graph architectures and agile processes (Angles, 2012; Francis et al., 2018).

Collectively, the findings suggest that the integration challenge is epistemological and procedural rather than technological.

5.2 Theoretical Contributions to System Analysis and Design

This paper is an addition to the System Analysis and Design (SAD) theory that revisits the implicit relational ontology of the theory. Decomposition, normalization, and schema stability are the key principles of modeling in traditional SAD (Codd, 1970; Wand & Weber, 1995) . These principles presuppose that the minimization of structural integrity and redundancy are the most universal design objectives. Nevertheless, in areas where connectivity is crucial, attribute normalization is not the only aspect of performance and expressiveness, as it relies on traversal semantics and relationship topology (Angles & Gutierrez, 2008; Burton-Jones et al., 2017).

The Graph-Agile Framework is a framework that promotes an abstraction of connections where relationships and traversal paths have been made first-order elements of analysis. According to this repositioning, normalization is represented as a context-dependent heuristic by context and no longer an imperative, which is in line with the graph-native performance features (Bonifati et al., 2018; Robinson et al., 2015). By doing so, the framework expands the discourse of the representation theory by preempting connectivity as a structural determinant of the system behavior (Burton-Jones et al., 2017).

The theoretical contribution is threefold:

1. **Reframing Normalization.**

The conceptualization of normalization is dependent on the nature of workload. Traversal-oriented denormalization could be a logical design decision instead of a departure of good modeling practices in a graph-based system (Bonifati et al., 2018; Robinson et al., 2015).

2. **Formalizing Traversal Awareness.**

Traversal semantics are institutionalized at the requirements stage, where the principles of value-driven design are applied in graph-centric contexts (Beck et al., 2001; Burton-Jones et al., 2017).

3. **Integrating Schema Optionality.**

Schema evolution is incorporated into the cyclical governance loop and is consistent with the way evolutionary database design and graph modeling should be practiced (S. Ambler & Sadalage, 2006; Holzschuher & Peinl, 2013).

According to the design science approach, the framework describes clear constructs, relationships, propositions, and boundary conditions, meeting the set standards of prescriptive theoretical contribution (Gregor & Hevner, 2013; Hevner et al., 2004; Jaakkola, 2020a).

5.3 Contributions to Agile Methodology Discourse

Agile software development emphasizes emergent design, iterative design, and minimal upfront architectural specification (Beck et al., 2001; Cockburn, 2006). Nevertheless, graph databases require early attention to the traversal depth, the directionality of the relationships, and the locality of performance (De Virgilio et al., 2014; Robinson et al., 2015).

The results suggest that emergent design and architectural foresight are not mutually exclusive but conditionally complementary when traversal constraints are clearly expressed during requirements engineering. By embedding traversal benchmarking and schema evolution into sprint rituals, the framework operationalizes agile principles of continuous improvement and technical excellence within graph-native contexts (Fowler, 2004; Schwaber & Sutherland, 2017).

Accordingly, the study contributes to agile scholarship by clarifying how domain-specific architectural constraints can be planned in a way that does not impair the iterative adaptability.

5.4 Interpretation of Empirical Evaluation

The quantitative results indicate consistently high expert agreement across dimensions of structural clarity, theoretical grounding, agile compatibility, support of scalability, and practical applicability. Such findings indicate that perceived internal coherence and conceptual validity are strong, which is also known in the literature of using expert judgement to evaluate design science artifacts (Jaakkola, 2020a; Whetten, 1989).

The relatively low, but yet high score of adoption feasibility provides analytical subtlety. Qualitative feedback explains this difference by the fact that it is expected that the learning curve will occur to graph-native modeling instead of structural incompatibility (Angles, 2012; Robinson et al., 2015) . This interpretation is supported by modeling template, training pathway and gradual implementation recommendations.

The results provide preliminary perceptual support for:

- **Proposition P1:** Sprint-level graph prototyping enhances scalability validation (Robinson et al., 2015).
- **Proposition P2:** Schema-optional graph modeling aligns with agile emergent design principles (Robinson et al., 2015).

Because the study relies on expert assessment rather than the longitudinal performance records, the findings are limited to the perceived coherence and methodology fit. As part of the rigor cycle of design science research (Gregor & Hevner, 2013; Hevner et al., 2004); the evaluations establish conceptual plausibility and practitioner relevance, while identifying empirical performance validation as a necessary next step.

5.5 Boundary Conditions and Contextual Applicability

The framework is best suited to:

- Systems that are relationship-intensive
- Network-based analytics
- Fraud detection platforms
- Knowledge graph architectures
- Recommendation systems

These contexts align with literature emphasizing graph database advantages in highly connected domains (Bonifati et al., 2018; Fensel et al., 2020; Newman, 2018). Conversely, in purely transactional or attribute-centric systems, traversal-oriented modeling may introduce unnecessary complexity, consistent with relational efficiency in structured workloads (Codd, 1970; Elmasri, Ramez and Navathe, 2016).

Explicit articulation of boundary conditions enhances theoretical precision and contextual validity.

5.6 Practical and Educational Implications

The framework gives practitioners a structured approach to creating scalable systems with a high degree of relationships within agile environments. It minimizes the risk of refactoring at late stages and performance deterioration by incorporating traversal-aware thinking on the strategic, tactical, and operational level, and being responsive to changing requirements (Iosup et al., 2015; Robinson et al., 2015). The integration of traversal benchmarking, query refinement, and controlled schema evolution into the sprint cycles leads to the reinforcement of an intentional architectural nature without the need to degrade agile adaptability, which makes scalability validation and performance optimization more normal governance practices than office reactions (Beck et al., 2001; Schwaber & Sutherland, 2017).

To system analysts, architects, and agile development teams, the framework provides a standard approach of converting business objectives into traversal-aware user stories and integrating connectivity-aware verification into iterative processes. This systematic integration assists in the early architectural foresight and maintains the flexibility that is the key to agile methods.

In case organizations are embracing the use of graph technologies, the structure acts as a guide towards the incremental adoption of graph databases within the current agile setting. The three-layer model offers the alignment of strategic goals and technical implementation allowing the teams to subsume graph-native practices with the existing agile workflows with few interruptions.

To the teachers and trainers, the framework brings forward new abilities in traversal-sensitive requirements engineering, schema-optional modelling and iterative graph validation. The competencies may be used to guide curriculum evolution of system analysis and design courses and to facilitate professional training programs to prepare analysts and architects to work in connectivity-intensive system environments.

5.7 Methodological Reflection and Future Research

The integration of design science research and theory synthesis has enabled the ability to combine streams of fragmented literatures into a coherent prescriptive artifact in a methodical manner (Gregor & Hevner, 2013; Hevner et al., 2004; Jaakkola, 2020a). The combined method assessment plan will add value to quantitative data through the triangulation based on the perception and qualitative remarks provided by the experts to increase the interpretive credibility. However, reliance on perceptual expert evaluation rather than longitudinal performance testing constitutes a primary limitation. Future research should incorporate:

- Industrial case studies and action research
- Comparative analyses between relational-agile and Graph-Agile Framework implementations
- Development and evaluation of Graph-Agile Framework
- Domain-specific empirical validation
- Stratified sampling to address expertise asymmetry

These investigations would make possible behavioral and scalability validation as compared to perceptual assessment.

6. CONCLUSION

This study addresses a structural gap in system analysis and design (SAD) by introducing the Graph-Agile Framework as a design science artifact that integrates graph database modeling with agile development practices. While relational modeling remains appropriate for many transactional contexts, its normalization-centric assumptions constrain expressiveness and scalability in connectivity-intensive systems. Concurrently, prevailing agile methodologies provide limited guidance for traversal-sensitive architectural reasoning in graph-native environments.

The proposed framework formalizes graph integration across three levels of mutually supporting nature, including Strategic Alignment, Tactical Integration, and Operational Adaptation, and this makes traversal awareness, schema optionality, and scalability validation within iterative development cycles. Rather than positioning graph modeling as a technical substitution for relational design, the framework reframes SAD toward a connectivity-centered abstraction in which traversal semantics become first-order analytical constructs.

The expert-based evaluation demonstrates high perceived conceptual clarity, theoretical grounding, agile compatibility, and scalability support. These findings provide preliminary perceptual validation of the framework's coherence and methodological fit, particularly regarding sprint-level traversal benchmarking and schema-evolution governance. However, conclusions remain bound to expert judgement; behavioral and performance-based validation is required to establish causal impact.

The study contributes theoretically by: re-examining the relational ontology underlying traditional SAD, reframing normalization as context-dependent rather than universally prescriptive, and institutionalizing traversal-aware reasoning within agile governance structures.

As a prescriptive conceptual artifact grounded in design science research (Gregor & Hevner, 2013; Hevner et al., 2004; Jaakkola, 2020a) ; the Graph-Agile Framework extends SAD toward connectivity-aware system development while preserving iterative adaptability. Future research should pursue longitudinal case studies, controlled comparative analyses, and performance benchmarking to evaluate behavioral and scalability outcomes across graph-native and relational-centric agile projects.

Through this integration, the study positions traversal semantics not merely as implementation concerns, but as foundational design constructs in the evolving theory and practice of scalable information systems.

REFERENCES

1. Aalabaf-Sabaghi, M. (2012). Networks, Crowds and Markets: Reasoning about a Highly Connected World. *Journal of the Royal Statistical Society Series A: Statistics in Society*, *175*(4). https://doi.org/10.1111/j.1467-985x.2012.01069_4.x
2. Ambler, S., & Sadalage, P. (2006). Refactoring databases: Evolutionary database design. In *Queue* (Vol. 4, Number 7).
3. Ambler, S. W. (2003). Agile Database Techniques — Effective Strategies for the Agile Software Developer. In *Database*.
4. Angles, R. (2012). A comparison of current graph database models. *Proceedings - 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW 2012*. https://doi.org/10.1109/ICDEW.2012.31
5. Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, *40*(1). https://doi.org/10.1145/1322432.1322433
6. Beck, K. (1999). Extreme Programming Explained: Embrace Change. In *XP Series*.
7. Beck, K., Beedle, M., Bennekum, A. Van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. The Agile Alliance.
8. Boehm, B. (2002). Get ready for agile methods, with care. *Computer*, *35*(1). https://doi.org/10.1109/2.976920
9. Bonifati, A., Fletcher, G., Voigt, H., & Yakovets, N. (2018). Querying Graphs. *Synthesis Lectures on Data Management*, *10*(3). https://doi.org/10.2200/s00873ed1v01y201808dtm051
10. Burton-Jones, A., Recker, J., Indulska, M., Green, P., & Weber, R. (2017). Assessing representation theory with a framework for pursuing success and failure. *MIS Quarterly: Management Information Systems*, *41*(4). https://doi.org/10.25300/MISQ/2017/41.4.13
11. Chen, P. P. S. (1976). The Entity-Relationship Model—toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, *1*(1). https://doi.org/10.1145/320434.320440
12. Cockburn, A. (2006). Agile software development: the cooperative game. *Building*, *113*.
13. Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, *13*(6). https://doi.org/10.1145/362384.362685
14. Cohn, M. (2004). User Stories Applied: For Agile Software Development (Addison Wesley Signature Series). In *Writing* (Vol. 1, Number 0).
15. Date, C. J. (2003). An Introduction to Database Systems 8e. In *Pearson*.
16. De Virgilio, R., Maccioni, A., & Torlone, R. (2014). Model-driven design of graph databases. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8824*. https://doi.org/10.1007/978-3-319-12206-9_14
17. Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. In *Journal of Systems and Software* (Vol. 85, Number 6). https://doi.org/10.1016/j.jss.2012.02.033
18. Dominguez-Sal, D., Urbón-Bayes, P., Giménez-Vañó, A., Gómez-Villamor, S., Martínez-Bazán, N., & Larriba-Pey, J. L. (2010). Survey of graph database performance on the HPC scalable graph analysis benchmark. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6185 LNCS*. https://doi.org/10.1007/978-3-642-16720-1_4
19. Dong, H., Dacre, N., Baxter, D., & Ceylan, S. (2024). What is Agile Project Management? Developing a New Definition Following a Systematic Literature Review. *Project Management Journal*, *55*(6). https://doi.org/10.1177/87569728241254095
20. Elmasri, Ramez and Navathe, S. B. (2016). Fundamentals of database systems seventh edition. In *Webseiten entwickeln mit ASP.NET*.
21. Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., & Wahler, A. (2020). Knowledge Graphs: Methodology, Tools and Selected Use Cases. In *Knowledge Graphs: Methodology, Tools and Selected Use Cases*. https://doi.org/10.1007/978-3-030-37439-6
22. Fowler, M. (2002). Refactoring: Improving the design of existing code. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2418*. https://doi.org/10.1007/3-540-45672-4_31
23. Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. *Pearson Paravia Bruno Mondad*.
24. Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., & Taylor, A. (2018). Cypher: An evolving query language for property graphs. *Proceedings of*

the ACM SIGMOD International Conference on Management of Data. https://doi.org/10.1145/3183713.3190657
25. Gregor, S. (2006). The nature of theory in Information Systems. *MIS Quarterly: Management Information Systems*, *30*(3). https://doi.org/10.2307/25148742
26. Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. In *MIS Quarterly: Management Information Systems* (Vol. 37, Number 2). https://doi.org/10.25300/MISQ/2013/37.2.01
27. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly: Management Information Systems*, *28*(1). https://doi.org/10.2307/25148625
28. Holzschuher, F., & Peinl, R. (2013). Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/2457317.2457351
29. Iosup, A., Hegeman, T., Ngai, W. L., Heldens, S., Pérez, A. P., Manhardt, T., Chafi, H., Capotă, M., Sundaram, N., Anderson, M., Tănase, I. G., Xia, Y., Nai, L., & Boncz, P. (2015). LDBC graphalytics: A benchmark for large scale graph analysis on parallel and distributed platforms. *Proceedings of the VLDB Endowment*, *9*(13). https://doi.org/10.14778/3007263.3007270
30. Jaakkola, E. (2020a). Designing conceptual articles: four approaches. *AMS Review*, *10*(1–2), 18–26. https://doi.org/10.1007/s13162-020-00161-0
31. Jaakkola, E. (2020b). Designing conceptual articles: four approaches. *AMS Review*, *10*(1–2), 18–26. https://doi.org/10.1007/s13162-020-00161-0
32. Nathan, A. J., & Scobell, A. (2017). OMG Unified Modeling Language, Version 2.5.1. *Foreign Affairs*, *91*(5).
33. Needham, M., & Hodler, A. E. (2020). Graph algorithms: Practical examples in Apache Spark & Neo4j. In *Angewandte Chemie International Edition, 6(11), 951–952.*
34. Newman, M. E. J. (2018). Network structure from rich but noisy data. In *Nature Physics* (Vol. 14, Number 6). https://doi.org/10.1038/s41567-018-0076-1
35. P. Abrahamson, Outi Salo, Jussi Ronkainen, & Juhani Warsta. (2002). Agile software development methods: Review and analysis. *VTT Publications*.
36. Pavlopoulos, G. A., Secrier, M., Moschopoulos, C. N., Soldatos, T. G., Kossida, S., Aerts, J., Schneider, R., & Bagos, P. G. (2011). Using graph theory to analyze biological networks. In *BioData Mining* (Vol. 4, Number 1). https://doi.org/10.1186/1756-0381-4-10
37. Rajaraman, A., & Ullman, J. D. (2011). Mining of massive datasets. In *Mining of Massive Datasets* (Vol. 9781107015357). https://doi.org/10.1017/CBO9781139058452
38. Robinson, I., Webber, J., & Eifrem, E. (2015). Graph Databases: New Opportunities for Connected Data. In *Information Management*.
39. Schwaber, K., & Sutherland, J. (2017). The Scrum Guide: The Definitive The Rules of the Game. *Scrum.Org and ScrumInc*, (November).
40. Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010). *A comparison of a graph database and a relational database*. https://doi.org/10.1145/1900008.1900067
41. Wand, Y., & Weber, R. (1995). On the deep structure of information systems. *Information Systems Journal*, *5*(3). https://doi.org/10.1111/j.1365-2575.1995.tb00108.x
42. Wand, Y., & Weber, R. (2002). Research commentary: Information systems and conceptual modeling - A research agenda. *Information Systems Research*, *13*(4). https://doi.org/10.1287/isre.13.4.363.69
43. Whetten, D. A. (1989). What Constitutes a Theoretical Contribution? *The Academy of Management Review*, *14*(4). https://doi.org/10.2307/258554