Topical and Logical Structure: A Comprehensive Methodology for Creating and Evolving Reference Architectures: [DRAFT]

Pouya Ataei

1 Introduction

Software architecture is a fundamental aspect of software-intensive systems, influencing their quality attributes and overall success Bass et al. (2012). Reference architectures, a specific type of software architecture, provide a template for architecting systems for a class of problems Angelov et al. (2012a). These architectures offer guidance for system development, standardization, and evolution Cloutier et al. (2010).

Despite the effectiveness of reference architecture to tackle complex problems Angelov et al. (2012c), there has not been much attention given to methodologies that help design and evolve reference architectures.

While there are various threads of knowoledge in academia to address architectural analysis, synthesis, and evaluation of reference architectures Dobrica and Niemela (2008); Galster et al. (2011); Muller and Laar (2008), they are not interwoven into a cohesive fabric that can be used to systematically create and evolve reference architectures. That is, there is a lack of a comprehensive methodology for creating and evolving reference architectures.

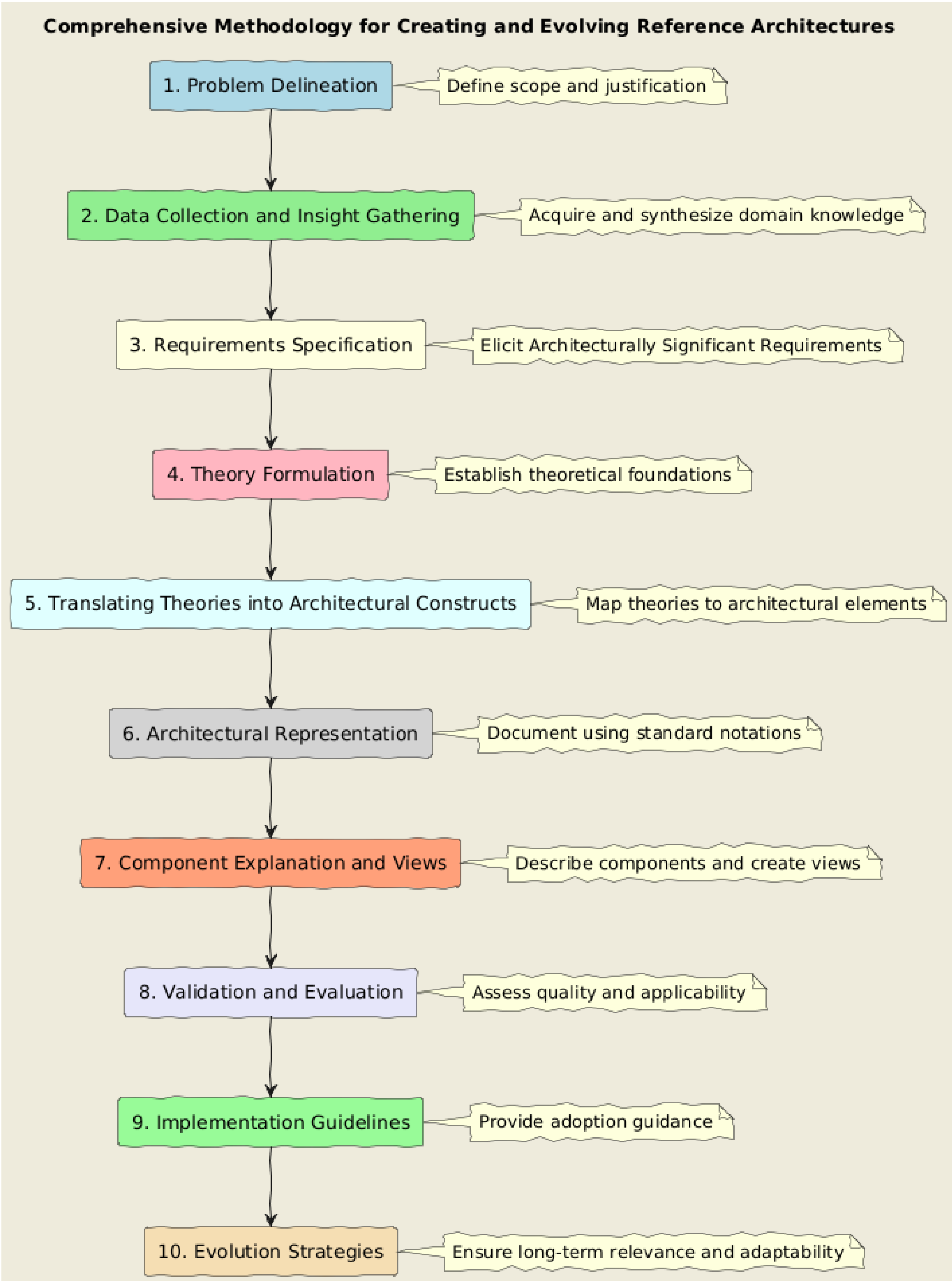
The absence of a unified approach to reference architectures has resulted in challenges for industry. These include interoperability issues when integrating systems from different vendors Weyrich and Ebert (2015), difficulties in standardizing software development processes Garcia-Moreno et al. (2020), and redundant development efforts due to the lack of a common architectural foundation Nakagawa et al. (2011). Furthermore, the lack of a clear methodology increases ambiguity for researchers aiming to create new reference architectures, potentially hindering progress in this field Angelov et al. (2012b). These challenges can lead to increased development costs and potential quality issues in software-intensive systems Antinyan (2020).

To address this gap, this study aims to provide a methodology for creating and evolving reference architectures.

The book is structured as follows: Chapter 2 provides an overview of reference architectures. Chapters 3 to 7 explore reference architectures in specific critical domains: telecommunications, healthcare, automotive, avionics, and Industry 4.0. Chapter 8 discusses domain-independent reference architectures and standards. Chapter 9 outlines future advances in reference architectures, and Chapter 10 presents the conclusions.

2 Related Work

- Concept: Existing methodologies for reference architecture creation
- Method: Critical analysis of current approaches, including Galster and Avgeriou (2011) and Nakagawa et al. (2014)
- Optional to consider: Analysis of domain-specific reference architectures (e.g., AUTOSAR, IIRA) (**?**)


Comprehensive Methodology for Creating and Evolving Reference Architectures
1. Problem Delineation
Define scope and justification
2. Data Collection and Insight Gathering
Acquire and synthesize domain knowledge
3. Requirements Specification
Elicit Architecturally Significant Requirements
4. Theory Formulation
Establish theoretical foundations
5. Translating Theories into Architectural Constructs
Map theories to architectural elements
6. Architectural Representation
Document using standard notations
7. Component Explanation and Views
Describe components and create views
8. Validation and Evaluation
Assess quality and applicability
9. Implementation Guidelines
Provide adoption guidance
10. Evolution Strategies
Ensure long-term relevance and adaptability

3 Problem Delineation

- Concept: Defining the scope and justification for the reference architecture
- Methods:
  - Systematic Literature Review (SLR) to identify gaps and challenges (?)
  - Multi-vocal Literature Review to capture practitioner perspectives (Garousi et al., 2019)
  - Stakeholder Analysis to understand diverse architectural needs (?)
  - Case Study Analysis to identify common architectural challenges (Runeson and Höst, 2009)
  - Gap Analysis to compare existing architectural approaches
  - Domain-Specific Metrics to quantify potential impact
- Outcomes:
  - Comprehensive problem statement based on academic and industry evidence
  - Quantified gaps in existing architectural approaches
  - Prioritized list of stakeholder needs and challenges
  - Clear justification for the proposed reference architecture
- Alignment with Design Science Research principles (Hevner et al., 2004)
- Consideration of reference architecture drivers (e.g., standardization, facilitation) (?)

4 Data Collection and Insight Gathering

- Concept: Comprehensive domain knowledge acquisition and synthesis
- Methods:
  - Systematic Literature Review (SLR) of academic sources (Kitchenham and Charters, 2007)
  - Multivocal Literature Review (MLR) of grey literature (Garousi et al., 2019)

  - Semi-structured expert interviews (Bogner et al., 2009)
  - Delphi study for expert consensus on domain challenges (?)
  - Ethnographic observations of domain practices (?)
  - Survey of domain practitioners (?)

- Analysis Techniques:
  - Thematic analysis of qualitative data (?)
  - Grounded theory for emerging concepts (?)
  - Cross-case analysis of existing systems (?)
  - Domain modeling using feature modeling or ontology engineering (?)

- Outcomes:
  - Comprehensive domain knowledge base
  - Identified patterns and trends in domain architectures
  - Catalog of stakeholder concerns and architectural drivers
  - Domain-specific challenges and opportunities for architectural innovation

- Considerations:
  - Analysis of existing systems and stakeholder concerns (?)
  - Integration of academic and practitioner perspectives
  - Identification of domain-specific quality attributes and constraints
  - Mapping of domain concepts to potential architectural elements

5 Requirements Specification

- Concept: Elicitation and documentation of Architecturally Significant Requirements (ASRs) (Chen et al., 2013)

- Methods:
  - Application of ISO/IEC/IEEE 29148:2018 standard (International Organization for Standardization, 2018)
  - Quality Attribute Workshops (QAW) to identify key quality attributes (?)

- Architecture Business Cycle (ABC) analysis to align with business goals (?)
- Utility Tree construction for prioritizing ASRs (?)
- Delphi technique for consensus on critical requirements (?)

- Outcomes:
  - Comprehensive set of ASRs with clear traceability to stakeholder needs
  - Prioritized list of quality attributes relevant to the reference architecture
  - Mapping of ASRs to architectural decisions and constraints
- Considerations:
  - Incorporation of domain-specific standards and regulations (?)
  - Analysis of variability in requirements across the domain (Galster et al., 2014)
  - Integration with model-based requirements engineering approaches (?)

6 Theory Formulation

- Concept: Establishing theoretical foundations for the reference architecture
- Methods:
  - Abductive inference for kernel and design theory development (Dubois and Gadde, 2014)
  - Grounded Theory approach for theory building from data (?)
  - Meta-ethnography for synthesizing qualitative studies (?)
  - Design Science Research for developing prescriptive theories (Gregor and Jones, 2013)
- Theory Development Process:
  - Identification of core concepts and relationships
  - Formulation of propositions and hypotheses

  - Development of explanatory and predictive models
  - Iterative refinement and validation of theories

- Theoretical Frameworks to Consider:
  - Architectural patterns and styles specific to the domain (?)
  - Contingency theory for context-dependent architectural decisions (?)
  - Systems theory for understanding complex interactions (?)
  - Socio-technical systems theory for aligning architecture with organizational context (?)

- Outcomes:
  - Kernel theories explaining fundamental domain principles
  - Design theories guiding architectural decision-making
  - Theoretical model of the reference architecture
  - Propositions for empirical validation

- Considerations:
  - Integration of domain-specific and general architectural theories
  - Alignment of theories with collected empirical data
  - Balancing explanatory power with practical applicability
  - Ensuring theoretical foundations support variability and evolution

7 Translating Theories into Architectural Constructs

- Concept: Systematic translation of theoretical foundations into concrete architectural elements

- Methods:
  - Theory-to-architecture mapping techniques (?)
  - Variability management approaches (Galster et al., 2014)
  - SPES modeling framework for model-based design (?)

  - Architecture Description Language (ADL) for formal representation (?)
  - Quality Attribute Scenarios for operationalizing quality requirements (?)

- Translation Process:
  - Identification of key theoretical concepts and relationships
  - Mapping of concepts to architectural elements and patterns
  - Definition of variability points and mechanisms
  - Formalization of architectural decisions and rationales
  - Integration of domain-specific constraints and standards

- Architectural Constructs to Consider:
  - Components, connectors, and their configurations
  - Architectural styles and patterns relevant to the domain
  - Variability mechanisms (e.g., parameterization, optional features)
  - Cross-cutting concerns and their architectural representations
  - Interfaces and protocols for inter-component communication

- Outcomes:
  - Comprehensive set of architectural constructs derived from theories
  - Formal architecture description using selected ADL
  - Variability model capturing architectural alternatives
  - Traceability links between theories and architectural elements
  - Set of architectural tactics addressing quality attributes

- Considerations:
  - Balancing abstraction and concreteness in architectural representations
  - Ensuring consistency between theoretical foundations and architectural constructs
  - Addressing domain-specific requirements and constraints
  - Facilitating extensibility and evolvability of the reference architecture
  - Validating the completeness and correctness of the translation

8 Architectural Representation

- Concept: Standardized and comprehensive documentation of the reference architecture
- Methods:
  - ISO/IEC/IEEE 42010:2011 for architecture description (International Organization for Standardization, 2011)
  - ArchiMate for enterprise architecture modeling (Lankhorst, 2017)
  - UML and SysML for system and software modeling (?)
  - Architecture Description Languages (ADLs) for formal representation (?)
  - Domain-specific modeling languages (e.g., AADL for embedded systems) (?)
  - Model-Based Systems Engineering (MBSE) approaches (?)
- Representation Process:
  - Identification of key stakeholders and their concerns (?)
  - Selection of appropriate viewpoints and views (Kruchten, 1995)
  - Definition of architecture elements and their relationships
  - Specification of interfaces and protocols
  - Documentation of architectural decisions and rationales (?)
  - Creation of architecture models using selected notations
- Representation Aspects to Consider:
  - Structural views (e.g., component diagrams, deployment diagrams)
  - Behavioral views (e.g., sequence diagrams, state machines)
  - Functional views (e.g., use case diagrams, activity diagrams)
  - Information views (e.g., data models, information flow diagrams)
  - Non-functional aspects (e.g., quality attribute scenarios, performance models)
  - Variability representation (e.g., feature models, variation points) (Galster et al., 2014)
- Outcomes:

- Comprehensive set of architecture views and models
- Formal architecture description using selected ADL or modeling language
- Traceability between architectural elements and stakeholder concerns
- Documentation of architectural patterns and styles used
- Representation of variability and extension points
- Alignment with domain-specific standards and best practices

• Considerations:

- Balancing detail and abstraction in architectural representations (?)
- Ensuring consistency across different views and models
- Addressing domain-specific representation requirements
- Facilitating communication among diverse stakeholders
- Supporting automated analysis and verification of architectural properties
- Enabling integration with model-driven development approaches (?)
- Consideration of emerging paradigms (e.g., IoT, Industry 4.0) in representation (?)

9 Component Explanation and Views

• Concept: Comprehensive architectural description through multiple perspectives

• Methods:

- 4+1 View Model of Architecture (Kruchten, 1995)
- Views and Beyond approach (?)
- Viewpoint-oriented systems engineering (?)
- ISO/IEC/IEEE 42010:2011 viewpoint framework (International Organization for Standardization, 2011)
- RAMI 4.0 viewpoints for industrial applications (?)

- View Types and Their Purpose:
  - Logical view: Functional requirements and system decomposition
  - Process view: Concurrency and synchronization aspects
  - Development view: Software management and reuse
  - Physical view: System topology and distribution
  - Scenarios: Integrating and validating the four views (Kruchten, 1995)
  - Information view: Data models and information flow (?)
  - Context view: System relationships and dependencies (?)
- Component Description Elements:
  - Interfaces and protocols
  - Behavior specifications
  - Quality attribute characteristics
  - Variability points and configuration options
  - Dependencies and constraints
  - Rationale for design decisions (?)
- View Integration and Consistency:
  - Cross-view traceability techniques (?)
  - Consistency checking methods (?)
  - View synchronization strategies (?)
- Outcomes:
  - Comprehensive set of architectural views
  - Detailed component descriptions with rationales
  - Traceability between views and stakeholder concerns
  - Consistency analysis results across views
  - Integration with domain-specific viewpoints (e.g., RAMI 4.0)
- Considerations:
  - Tailoring views to specific stakeholder needs (?)
  - Balancing completeness with understandability

  - Addressing domain-specific view requirements
  - Integrating with model-driven approaches (?)
  - Supporting architectural knowledge management (?)
  - Facilitating architecture evaluation through views (?)
  - Consideration of emerging paradigms (e.g., IoT, Industry 4.0) in viewpoint selection (?)

10 Validation and Evaluation

- Concept: Rigorous quality assessment and validation of the reference architecture
- Methods:
  - Case studies for real-world application assessment (Runeson and Höst, 2009)
  - Expert evaluations and surveys (Beecham et al., 2005)
  - Simulation and modeling for performance analysis (Martens et al., 2010)
  - Architecture Tradeoff Analysis Method (ATAM) (?)
  - Scenario-based architecture analysis (?)
  - Prototype implementation and testing (?)
  - Formal verification techniques (?)
- Evaluation Criteria:
  - Functional correctness and completeness
  - Quality attribute satisfaction (e.g., performance, security, maintainability)
  - Stakeholder concern coverage
  - Architectural style and pattern appropriateness
  - Variability and extensibility support
  - Compliance with domain-specific standards and regulations (?)
  - Interoperability and integration capabilities
- Validation Process:

  - Definition of validation goals and metrics
  - Selection of appropriate validation methods
  - Design and execution of validation experiments
  - Data collection and analysis
  - Interpretation of results and feedback incorporation
  - Iterative refinement of the reference architecture

- Outcomes:

  - Quantitative and qualitative assessment results
  - Identified strengths and weaknesses of the reference architecture
  - Validation reports and documentation
  - Recommendations for architecture improvements
  - Confidence level in the architecture's applicability and effectiveness

- Considerations:

  - Balancing thoroughness of evaluation with time and resource constraints
  - Addressing domain-specific validation requirements
  - Ensuring objectivity and reducing bias in expert evaluations
  - Validating both structural and behavioral aspects of the architecture
  - Assessing the architecture's ability to meet future domain challenges
  - Evaluating the architecture's support for emerging technologies and paradigms (**?**)
  - Considering the impact of architectural decisions on system quality attributes (**?**)

11 Implementation Guidelines

- Concept: Bridging theory and practice in reference architecture adoption

- Methods:

- Detailed guidance based on common implementation challenges (Martínez-Fernández et al., 2013)
- Architectural instantiation processes (Angelov et al., 2012b)
- Tailoring strategies for specific organizational contexts (Galster et al., 2014)
- Pattern-based architecture realization (?)
- Model-driven architecture implementation approaches (?)

- Implementation Process:
  - Gap analysis between current and target architecture (?)
  - Prioritization of implementation activities (?)
  - Incremental adoption strategies (?)
  - Customization and extension of reference architecture components (Galster and Avgeriou, 2011)
  - Integration with existing systems and processes (Lankhorst, 2017)

- Key Considerations:
  - Alignment with business goals and stakeholder requirements (?)
  - Handling of architectural variability points (Galster et al., 2014)
  - Management of architectural constraints and trade-offs (?)
  - Consideration of non-functional requirements in implementation (?)
  - Addressing organizational and cultural challenges (?)

- Outcomes:
  - Detailed implementation roadmap
  - Customized reference architecture instances
  - Set of best practices and lessons learned
  - Guidelines for architectural governance during implementation
  - Metrics for measuring implementation success

- Challenges and Mitigation Strategies:
  - Overcoming resistance to architectural change (?)
  - Managing complexity in large-scale implementations (?)

  - Ensuring consistency across different implementation projects (?)
  - Balancing standardization with flexibility (Angelov et al., 2012b)
  - Addressing skills gaps and training needs (Martínez-Fernández et al., 2013)

- Emerging Trends:
  - Agile and iterative implementation approaches (?)
  - DevOps integration in architecture implementation (?)
  - Consideration of emerging technologies (e.g., microservices, containerization) (?)
  - Adaptation to Industry 4.0 and IoT paradigms (?)

12 Evolution Strategies

- Concept: Ensuring long-term relevance and adaptability of the reference architecture

- Methods:
  - Continuous refinement techniques and adaptation mechanisms (Eixelsberger et al., 1998)
  - Architecture-centric evolution approaches (?)
  - Change impact analysis methods (?)
  - Version control and configuration management for architectures (?)
  - Architectural knowledge management for evolution support (?)

- Evolution Process:
  - Periodic architecture assessments and gap analysis (?)
  - Identification of architectural drift and erosion (?)
  - Prioritization of evolution needs based on stakeholder feedback (?)
  - Incremental and iterative architecture updates (?)
  - Documentation and communication of architectural changes (?)

- Key Considerations:

  - Balancing stability and flexibility in the architecture (?)
  - Managing architectural technical debt (?)
  - Ensuring backward compatibility during evolution (?)
  - Adapting to emerging technologies and paradigms (?)
  - Maintaining traceability between evolving architectural elements (?)

- Evolution Strategies:
  - Modularization and loose coupling for easier component updates (?)
  - Design for variability and extensibility (Galster et al., 2014)
  - Use of architectural patterns that support evolution (?)
  - Adoption of microservices for independent service evolution (?)
  - Implementation of feature toggles for gradual feature introduction (?)

- Outcomes:
  - Evolving reference architecture that remains relevant over time
  - Documented evolution history and rationale
  - Set of evolution patterns and best practices
  - Metrics for measuring architecture evolvability
  - Reduced architectural technical debt

- Challenges and Mitigation:
  - Managing complexity during long-term evolution (?)
  - Balancing short-term needs with long-term architectural integrity (?)
  - Ensuring consistency across different versions of the architecture (?)
  - Addressing resistance to architectural changes (?)
  - Maintaining architectural knowledge throughout evolution (?)

13 Threats to Validity

- Concept: Methodology limitations and potential biases
- Method: Systematic identification and mitigation strategies (Wohlin et al., 2012)
- Optional to consider: Consideration of domain-specific challenges and limitations (?)

14 Discussion

- Concept: Comparative analysis and potential impact of the proposed methodology
- Method: Critical reflection on methodology strengths and limitations
- Optional to consider: Discussion on the role of reference architectures in emerging paradigms (e.g., IoT, Industry 4.0) (?)

15 Conclusion

- Concept: Synthesis of contributions to reference architecture design
- Method: Summary of key methodological advancements and future research directions
- Optional to consider: Reflection on the future of reference architectures and their role in system development (?)

References

Angelov, S., Grefen, P., and Greefhorst, D. (2012a). A framework for analysis and design of software reference architectures. *Information and Software Technology*, 54(4):417–431.

Angelov, S., Grefen, P., and Greefhorst, D. (2012b). A framework for analysis and design of software reference architectures. *Information and Software Technology*, 54(4):417–431.

Angelov, S., Trienekens, J. J., and Grefen, P. (2012c). Designing and applying an approach to software reference architecture evaluation. *Information and Software Technology*, 54(9):1012–1028.

Antinyan, V. (2020). Revealing the complexity of automotive software. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1525–1528.

Bass, L., Clements, P., and Kazman, R. (2012). *Software architecture in practice*. Addison-Wesley Professional.

Beecham, S., Hall, T., and Rainer, A. (2005). Using an expert panel to validate a requirements process improvement model. *Journal of Systems and Software*, 76(3):251–275.

Bogner, A., Littig, B., and Menz, W. (2009). *Interviewing experts*. Springer.

Chen, L., Ali Babar, M., and Nuseibeh, B. (2013). Characterizing architecturally significant requirements. *IEEE software*, 30(2):38–45.

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., and Bone, M. (2010). The concept of reference architectures. *Systems Engineering*, 13(1):14–27.

Dobrica, L. and Niemela, E. (2008). An approach to reference architecture design for different domains of embedded systems. In *2008 International Conference on Software Engineering Research & Practice*, pages 287–293.

Dubois, A. and Gadde, L.-E. (2014). Abductive reasoning in logistics research. *International Journal of Physical Distribution & Logistics Management*.

Eixelsberger, W., Ogris, M., Gall, H., and Bellay, B. (1998). Software architecture reconstruction: Practice needs and current approaches. *IEEE Transactions on Software Engineering*, 24(9):797–812.

Galster, M. and Avgeriou, P. (2011). Empirically-grounded reference architectures: a proposal. In *Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS*, pages 153–158.

Galster, M., Avgeriou, P., Weyns, D., and Mannisto, T. (2011). Empirically-grounded reference architectures: a proposal. In *Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS*, pages 153–158.

Galster, M., Weyns, D., Tofan, D., Michalik, B., and Avgeriou, P. (2014). Variability in software systems—a systematic literature review. *IEEE Transactions on Software Engineering*, 40(3):282–306.

Garcia-Moreno, F. M., Bermudez-Edo, M., Garrido, J. L., Rodríguez-García, E., Pérez-Mármol, J. M., and Rodríguez-Fórtiz, M. J. (2020). A microservices e-health system for ecological frailty assessment using wearables. *Sensors*, 20(12):3427.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106:101–121.

Gregor, S. and Jones, D. (2013). The anatomy of a design theory. *Journal of the Association for Information Systems*, 8(5):312–335.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

International Organization for Standardization (2011). Iso/iec/ieee 42010: 2011-systems and software engineering–architecture description.

International Organization for Standardization (2018). Iso/iec/ieee 29148: 2018 systems and software engineering—life cycle processes—requirements engineering.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, 12(6):42–50.

Lankhorst, M. (2017). *Enterprise architecture at work*. Springer.

Martens, A., Koziolek, H., Becker, S., and Reussner, R. (2010). Automated evaluation of reference architectures: A case study. *Central European Journal of Computer Science*, 1(1):100–116.

Martínez-Fernández, S., Ayala, C. P., Franch, X., and Marques, H. M. (2013). Applying and evaluating reference architectures in a software factory. *International Journal of Software Engineering and Knowledge Engineering*, 23(09):1267–1297.

Muller, G. and Laar, P. v. d. (2008). Right sizing reference architectures-how to provide specific guidance with limited information. In *INCOSE International Symposium*, volume 18, pages 2047–2058. Wiley Online Library.

Nakagawa, E. Y., Ferrari, F. C., Sasaki, M. M. F., and Maldonado, J. C. (2011). An aspect-oriented reference architecture for software engineering environments. *Journal of systems and software*, 84(10):1670–1684.

Nakagawa, E. Y., Oquendo, F., and Becker, M. (2014). A process to create reference architectures for software ecosystems. In *European Conference on Software Architecture*, pages 320–337. Springer.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131.

Weyrich, M. and Ebert, C. (2015). Reference architectures for the internet of things. *IEEE software*, 33(1):112–116.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.