A Review of Software Architecture Evaluation Methods for Sustainability Assessment

Iffat Fatima, Patricia Lago *Vrije Universiteit Amsterdam*
Amsterdam, The Netherlands
i.fatima@vu.nl, p.lago@vu.nl

Appendix A
Results

Table I shows the evaluation techniques used by Software Architecture (SA) evaluation methods and their definitions. Overall, Fig. 1 shows the publication trend over the course of time. Although no year limit was applied in the search results, no relevant primary study was found before 1994 or after 2020. Fig. 2 shows the distribution of studies by method of their inception or their application. Only 15 methods (22.1%, see reftable:tools) provide tool support for SA evaluation. SA evaluation methods in [2], [3] use third-party tools for certain steps of the evaluation process while others automate the whole evaluation process through tool usage.

To classify the QAs (covered in the studies) according to the 4D-sustainability, we use the ISO/IEC 25010 standard [1]: where possible, we map the QA definitions on the existing sub-characteristics (see the second column in Table IV).

TABLE I: Evaluation techniques used by Architecture Evaluation Analysis Methods and their definitions

Evaluation Technique	**Description**
Scenario-based	Evaluation is based on the description of a functional scenario or use case of the system in order to assess its quality. The scenario can be represented as a scenario profile, UML diagram, textual description
Metric-based	Evaluation is based on a set of pre-defined metrics that are quantified to assess the quality of SA
Experience-based	Evaluation is based on the experience of experts based on system and domain-specific requirements
Checklist-based	Evaluation is based on conformance to a predefined assessment checklist
Model-based	Evaluation is based on usage of a quality model
Simulation-based	Evaluation is based on the simulation results of a proposed SA. Simulations may also provide quantification of certain QAs
Mathematical-modeling based	System is represented as a mathematical model and quality is assessed from quality functions with pre-defined threshold values
Static Architecture Evaluation based	Evaluation is based on comparing the planned architectural model with the source code model
Test-based	Evaluation is based on the results of a regression test suite on the SA representation of the system
GQM-based	Goals are based on the expected attributes of the architecture. Questions are used to identify the characteristics of architecture. Metrics aid in providing a quantitative assessment.
Visualization-based	Evaluation is based on the visual representation of component dependencies as a graph
Formal-model-based	Evaluation is based on a formal model based on SA characteristics and evaluation heuristics, in formal language notation

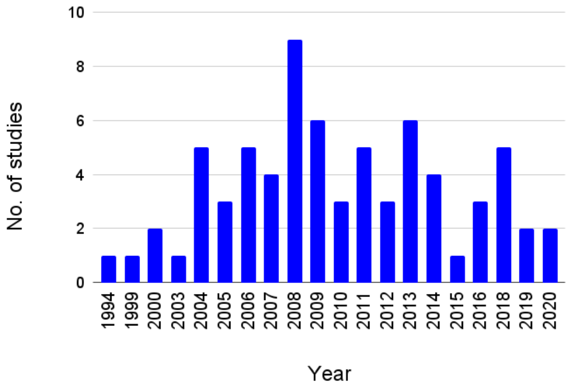

Fig. 1: Total studies published over the years for SA evaluation methods

[1] https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

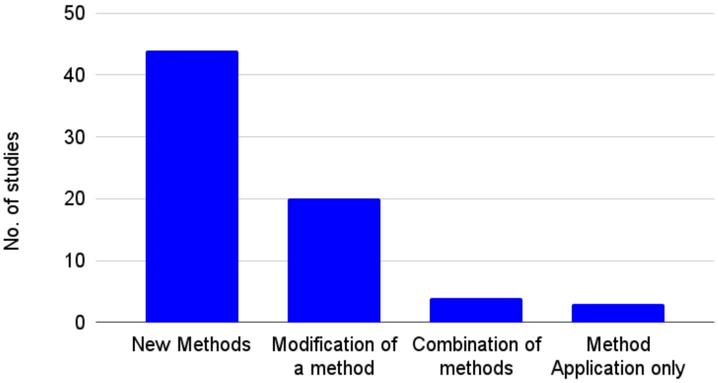

Fig. 2: Total studies and the type of inception/application of methods

TABLE II: Tool support provided by SA evaluation methods

Tools	Study ID	Description
Argus-I toolkit	S2	A specification-based tool to perform analysis and testing (Component and architectural levels)
SAVE tool	S13	Eclipse Plugin to understand the relationship between architectural model and source code
SQUARE	S1	Software QUality and ARchitecture modeling Environment is a tool for software quality modeling and analysis. It uses the HASARD (Hazard Analysis of Software ARchitectural Designs) method for automated analysis of quality models
SARA Tool	S31	For model-based risk assessment that uses data from domain experts and UML-based measures at design phase
UML-JMT	S37	For automated assessment of performance characteristics using a UML model of architecture. The evaluation is performed by conversion of UML model to EQN (Extended Queuing Network) performance model
LSTA Tool	S42	For implied scenario analysis. It uses system representation as Message Sequence Charts
CSAFE Toolset	S43	Converts UML design to XMI specification for extracting architectural properties. It also allows the specification of scenarios and a mechanism for their refinement
Un-named	S46	Study states tool usage for automating the analysis process, however details of tool used are missing
Magnify	S47	Generates graph-based visualization of dependencies at different levels (i.e. package level, class level, etc.) using source code
CORE Tool	S52	It accepts formal model axioms to model as filters in CORE tool for ease of selection for the user
SD Metrics	S54	For analysis of software design
OpManager	S54	For monitoring software performance
Altova UModel	S54	For generating diagrams
Simulator Plugin	S57	A tool responsible for the interpretation of execution traces of a Go program
Checker plugin	S57	A tool to verify if ADL-based architectural model fulfills properties defined using DynBLTL
SPARTA	S58	It uses DFD models for threat modeling. It imports catalogs for privacy and security solutions. It also allows building an attacker model to run risk analysis on DFD
CPN Tool	S60	For creating colored petri-nets and simulating them
DEVS-SUITE	S67	To simulate SA using discrete event system specification to build a simulation environment

TABLE III: QAs for SA quality and sustainability mapping

QAs	Study ID	Ec	E	S	T
Applicability	S22				●
Buildability	S22, S17				●
Conceptual Integrity	S22				●
Complexity	S26, S3				●
Variability	S26				●
Similarity	S26, S59				●
Architectural Integrity	S23, S28, S45				●
Design Size	S61, S5	○	○	○	●
Hierarchies	S61, S59	○	○	○	●
Abstraction	S61, S59	○	○		●
Encapsulation	S61, S59	○	○	○	●
Coupling	S41, S44, S61, S59	○	○	○	●
Cohesion	S41, S44, S61, S59	○	○	○	●
Composition	S61, S59				●
Inheritance	S61, S59			○	●
Polymorphism	S61, S59	○	○	○	●
Messaging	S61, S59	○			●
Complexity	S61, S59	○	○		●

○ = potential mapping, ● = mapping

TABLE IV: 4D-sustainability mapping of QAs and their sub-characteristics for the supported systems

QAs	Sub-characteristics	Studies	Ec	E	S	T
Functional Suitability		S66	●			●
	Component Adequacy	S13				●
	Functional Completeness	S22				●
Performance Efficiency		S12, S27, S54, S66, S1, S16, S17, S34, S34, S37, S49, S50, S62, S66, S67	●	●		●
	Time behaviour	S50, S65		●		●
	Battery Utilization	S65		●		
	Energy Efficiency	S55		●		
	Memory Usage	S50		●		
Usability		S10, S12, S16, S17, S22, S25, S27, S32, S34, S34, S66	●		●	
	Learnability	S12, S27, S66	●		●	
	User Interface Aesthetics	S12, S27			●	
	Accessibility	S66			●	
Reliability		S27, S30, S11, S12, S43, S54, S63, S65, S66, S67				●
	Availability	S1, S11, S17, S34, S60, S62, S66, S67	●	●	●	●
	Freedom from risk	S3, S66	●		●	
	Robustness	S66				●
	Recoverability	S21	●			●
Security		S1, S11, S17, S34, S38, S39, S40, S54, S58, S62, S66			●	●
	Privacy	S58, S66			●	
	Confidentiality	S59, S66			●	
	Integrity	S34, S50, S59, S66			●	●
	Non-repudiation	S59			●	●
	Accountability	S59, S66			●	
	Authenticity	S59, S66			●	
Maintainability		S4, S9, S34, S43, S46, S56, S61, S66	●	●		●
	Evolvability	S13, S23, S28, S45				●
	Integratability	S54				●
	Modularity	S18, S44, S56, S61, S66				●
	Reusability	S26, S44, S54, S56, S61, S66	●	●		
	Analysability	S23, S28, S45, S56, S61				●
	Modifiability	S1, S5, S11, S24, S34, S34, S44, S54, S56, S61, S62, S66, S69, S71	●	●		●
	Changeability	S23, S28, S45	●			●
	Extensibility	S17, S23, S24, S28, S45	●			●
	Testability	S11, S23, S28, S34, S44, S45, S56, S61, S66				●
Portability		S23, S24, S28, S43, S45, S54, S66	●			●
	Adaptability	S11, S29				●
	Scalability	S17, S24, S54, S66	●			
	Survivability	S66				●
Other QAs	Cost	S11, S65	●			
	Time-to-market	S11	●			
	Persistence	S50				●
	Safety	S8			●	
	Context Coverage	S66				●
	Effectiveness	S66	●	●	●	●
	Satisfaction (Trust)	S66	●		●	●

Appendix B
Threats to Validity

According to Ampatzoglou et al. [1], there are no threats-to-validity types specific for SLRs. However, construct validity, reliability, internal- and external validity are commonly used. We follow this trend.

Construct Validity: Our review identifies a total of 71 studies will all major methods identified in other secondary studies in Related Work. The inclusion and exclusion criteria were applied on the first 1000 results of Google Scholar as the search engine only shows the first 1000 results to the users[2]. However, these articles are shown based on their relevance to the search query[3]. To ensure the extraction of a maximum number of relevant studies, they were extracted in clusters of 100. For SQ1, no relevant study was found in the last 150 studies by title checking hence we can reasonably assume that relevant and sufficient studies were extracted for analysis by these 1000 results. It is possible that some relevant studies were not included in our search results due to Google Scholar's delimitation to 1000 results. To mitigate this issue we ran a second search query, SQ2 to make sure no sustainability-specific studies were excluded during the initial search process. A total of 185 results (out of 300) of SQ2 were already found in SQ1 search results and hence, excluded. A majority (113 studies) of the remaining studies, from SQ2, were excluded as they were unrelated to our research. We also used wildcard character (*) and multiple synonym terms for "evaluation" to prevent missing any relevant studies. Moreover, the search query was run on full-text to avoid the exclusion of relevant studies. Studies whose titles were not clear for inclusion were evaluated through abstract checking, instead of exclusion. Similarly, full-text analysis was performed for studies whose abstracts lacked clarity. All major methods presented by studies discussed as related work were represented in our SLR.

[2] https://scholar.google.com/intl/en/scholar/help.html#export

[3] https://scholar.google.com/intl/en/scholar/help.html#overview

Reliability: In order to ensure the correctness of results, a systematic approach was used for data collection, categorization and analysis. Bias in the synthesis of results was possibly removed by reaching a consensus on decisions with the second author. Moreover, a systematic approach was used to keep the study as independent as possible of subjective bias. In order to replicate the results of the study, we provide a replication package that can be used to replicate or extend this study.

Internal Validity: There is a possibility of internal bias in the selection of studies. However, we defined a protocol for our methodology through inclusion and exclusion criteria. The mapping of QAs with sustainability dimensions was done based on their definitions. However, it is possible that one QA may fall additionally in another dimension as that depends on context. We mitigated this issue by cross-checking the mapping with studies that evaluated SA for sustainability. Moreover, all results and the process for the synthesis was cross-validated by the coauthor of the paper.

External Validity: We do not restrict our selection process to time or meta-data in order to find as many relevant primary studies for SA evaluation. Moreover, the synthesis of our results for RQ1 is not restricted to any context. Hence, the findings of this study can be used for the application, development or improvement of SA evaluation methods in general as well. We also represent this by mapping our findings of RQ1 for sustainability support in SA evaluation methods in general.

TABLE V: PRIMARY STUDIES

Study ID	Title	Year
S1	Experience with performing architecture tradeoff analysis	1999
S2	Software architecture analysis based on statechart semantics	2000
S3	A methodology for architectural-level risk assessment using dynamic metrics	2000
S4	An Empirically-Based Process for Software Architecture Evaluation	2003
S5	Architecture-level modifiability analysis (ALMA)	2004
S6	A scenario-driven approach for value, risk, and cost analysis in system architecting for innovation	2004
S7	ASAAM: aspectual software architecture analysis method	2004
S8	Knowledge centered assessment pattern: an effective tool for assessing safety concerns in software architecture	2004
S9	Evaluating an embedded software reference architecture-industrial experience report	2005
S10	Case studies on analyzing software architectures for usability	2005
S11	Information System Architecture Evaluation: From Software to Enterprise Level Approaches	2005
S12	Software Architecture Analysis of Usability	2004
S13	Static evaluation of software architectures	2006
S14	Applying dynamic change impact analysis in component-based architecture design	2006
S15	The essential components of software architecture design and analysis	2006
S16	Tool support to model-based quality analysis of software architecture	2006
S17	MEMS: A Method for Evaluating Middleware Architectures	2006
S18	On the Modularity of Software Architectures: A Concern-Driven Measurement Framework	2007
S19	Beyond ATAM: Architecture Analysis in the Development of Large Scale Software Systems	2007
S20	A holistic architecture assessment method for software product lines	2007
S21	Extending Failure Modes and Effects Analysis Approach for Reliability Analysis at the Software Architecture Design Level	2007
S22	Towards a method for the evaluation of reference architectures: Experiences from a case	2008
S23	Using dependency model to support software architecture evolution	2008
S24	Extending ATAM to assess product line architecture	2008
S25	Comprehensive Architecture Evaluation and Management in Large Software-Systems	2008
S26	Some Metrics for Accessing Quality of Product Line Architecture	2008
S27	Experiences with Software Architecture Analysis of Usability	2008
S28	Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study	2008
S29	Adaptability Evaluation at Software Architecture Level	2008
S30	Software architecture reliability analysis using failure scenarios	2008
S31	Using software architecture risk assessment for product line architectures	2009
S32	Towards a Method for Analyzing Architectural Support Levels of Usability	2009
S33	A Framework for Supporting the Software Architecture Evaluation Process in Global Software Development	2009
S34	Software architecture evaluation methods based on cost benefit analysis and quantitative decision making	2009

continued on next page

TABLE V: PRIMARY STUDIES

Study ID	Title	Year
S35	A new AHP-based approach towards Enterprise Architecture quality attribute analysis	2009
S36	Lightweight and continuous architectural software quality assurance using the aSQA technique	2010
S37	Cpasa: continuous performance assessment of software architecture	2010
S38	A scenario-based framework for the security evaluation of software architecture	2010
S39	Security risk analysis of software architecture based on AHP	2011
S40	Security Risk Assessment of Software Architecture	2011
S41	Assessing legacy software architecture with the autonomy ratio metric	2011
S42	Evaluating security properties of architectures in unpredictable environments: A case for cloud	2011
S43	An Architecture Analysis Approach for Supporting Black-Box Software Development	2011
S44	MORPHOSIS: A lightweight method facilitating sustainable software architectures	2012
S45	Software architecture evolution through evolvability analysis	2012
S46	An Approach towards Enterprise Architecture Analysis using AHP and Fuzzy AHP	2012
S47	On quick comprehension and assessment of software	2013
S48	On the Need of a Methodological Approach for the Assessment of Software Architectures within ISO26262	2013
S49	Early performance assessment in component-based software systems	2013
S50	Enterprise Information Systems Architecture—Analysis and Evaluation	2013
S51	Scenario-driven architecture assessment methodology for large data analysis systems	2013
S52	A Formal Method for Evaluation of a Modeled System Architecture	2013
S53	Towards quantitative metrics for architecture models	2014
S54	Software with service oriented architecture quality assessment	2014
S55	Model-based energy efficiency analysis of software architectures	2015
S56	An empirical evaluation model for software architecture maintainability for object oriented design	2016
S57	Statistical model checking of dynamic software architectures	2016
S58	SPARTA: Security & Privacy Architecture Through Risk-Driven Threat Assessment	2018
S59	Software Architectural Quality Assessment Model for Security Analysis Using Fuzzy Analytical Hierarchy Process (FAHP) Method	2018
S60	Availability Assessment of Software Systems Architecture Using Formal Models	2018
S61	A Fuzzy Analytical Hierarchy Process (FAHP) Based Software Quality Assessment Model: Maintainability Analysis	2018
S62	Assessing migration of a 20-year-old system to a micro-service platform using ATAM	2019
S63	Early reliability assessment of component-based software system using colored petri net	2019
S64	Quick Evaluation of a Software Architecture Using the Decision-Centric Architecture Review Method: An Experience Report	2020
S65	Sustainability Debt: A Portfolio-Based Approach for Evaluating Sustainability Requirements in Architectures	2016
S66	An Action Research for Improving the Sustainability Assessment Framework Instruments	2020
S67	Modeling and simulation of software architecture in discrete event system specification for quality evaluation	2014
S68	A Fuzzy AHP Based Approach Towards Enterprise Architecture Evaluation	2009
S69	SAAM: a method for analyzing the properties of software architectures	1994
S70	Lightweight evaluation of software architecture decisions	2014
S71	Enterprise Architecture Modifiability Analysis	2018

References

[1] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, and A. Chatzigeorgiou, "Identifying, categorizing and mitigating threats to validity in software engineering secondary studies," *Information and Software Technology*, vol. 106, pp. 201–230, 2019.

[2] A. Nuraini and Y. Widyani, "Software with service oriented architecture quality assessment," in *2014 International Conference on Data and Software Engineering (ICODSE)*, 2014, pp. 1–6.

[3] M. Sadeghian and H. Motameni, "Availability Assessment of Software Systems Architecture Using Formal Models," *International Journal of Computer Applications Technology and Research*, pp. 193–199, 2018.