A Systematic Mapping on Visual Solutions to Support the Comprehension of Software Architecture Evolution

Joao Werther[1], Glauco de Figueiredo Carneiro[2], Rita Suzana Pitangueira Maciel[1]
[1]Federal University of Bahia, BRA
[2]Universidade Salvador (UNIFACS), BRA
jwertherf@gmail.com, glauco.carneiro@unifacs.br, ritasuzana@dcc.ufba.br

Abstract

Context: Software visualization has the potential to support specialized stakeholders to understand the software architecture (SA) evolution. To the best of our knowledge, there is no guideline based on which visual solutions should be applied to support the SA evolution comprehension and how to use them. Goal: Analyze the use of visual solutions for the purpose of comprehension with respect to software architecture evolution from the point of view of software architects and developers in the context of both academia and industry. Method: We conducted a Systematic Mapping Study to achieve the stated goal. Results: The study identified 92 papers published from January 2000 to December 2018 as a result of the search strings execution. We selected 12 primary studies and identified a gap in terms of a taxonomy to assist specialists in the development or classification of solutions to support the comprehension of software architecture evolution using visual resources. Conclusion: We observed that despite the relevance of the use of visual solutions to support the comprehension of software architecture evolution, only 12 studies have reported these initiatives, suggesting that there is still room for the use of different visual metaphors to represent its components, relationships and evolution throughout the releases.

1. Introduction

Software evolution reflects changes undergone by the software during its lifespan [1]. The study of software evolution is essential to better support changes in software requirements over time, keeping its integrity at a lowest possible cost [2][3][4]. Software Architecture (SA) is the design model used to build and evolve a software system [5]. Throughout the analysis of the software architecture, in high level structure, it is possible to understand what dimensions along which a system is expected to evolve [1]. The importance of SA in software evolution process is that a software built without an adaptable architecture normally will degenerate sooner than others with a change-ready architecture [1]. The evolution of a SA can be the result of changes in the current SA to accommodate business demands, new technologies and/or platform or other reason that impacts the SA [6]. The comprehension of SA is essential for development and evolution of software systems [7][8] and can be supported by software visualization (SV) resources to understand key SA characteristics regarding architectural models and design decisions [7] [9].

To the best of our knowledge, there is no guideline to support the use of visual solutions towards SA evolution comprehension. For this reason, we conducted a Systematic Mapping Study (SMS) to identify studies that focus on this issue based on evidence provided by papers published in peer-reviewed conferences and journals from January 2000 to December 2018. We initially found 92 papers as a result of the applied search strings in specific electronic databases, from which we considered 12 studies as relevant.

This paper is organized as follows. Section 2 discusses related works and Section 3 presents the protocol of this SM. The Section 4 presents the main results of this study, also introducing the taxonomy proposed for visual solutions to software architecture evolution and presenting the main characteristics and features for each visual solution. The Section 5 discusses about the answers to research questions. Finally, Section 6 gives the conclusions of this work and suggests future works.

2. Related Works

Software visualization has been used in different areas of software engineering such as software architecture, software evolution and software design [7][10]. In the following paragraphs we highlight the importance of specific secondary studies that discussed the use of software visualization to support the execution of activities targeting software architecture. This is not an exhaustive list, it is rather an illustrative set of relevant papers whose discussion motivated the conduction of this systematic mapping.

Shahin, Liang and Babar [7] conducted a systematic review to characterize how Visualization Techniques (VT) have been used to represent SA in different application domains. Results classified the VTs into four types, according to its popularity: graph-based, notation-based, matrix-based and metaphor-based, where graph-based and notation-based VTs appear to be the most popular ones. However, the graph-based was pointed as the most popular architecture visualization technique in industry, largely employed. They

DOI reference number: 10.18293/DMSVIVA2019-008

concluded that VTs have been used to support SA activities for several purposes and highlighted the following as relevant: (i) the understanding of architecture evolution; (ii) the understanding of static characteristics of architecture; and (iii) search, navigation, and exploration of architecture design [7]. Additionally, conclusion highlighted that VTs have been applied to SA in a large range of domains, highlighting "graphics software" and "distributed system" as those that have been received the most attention in industry. Finally, in the conclusion section, the authors argued that SV is one of the increasingly popular ways to support the understanding the rationale behind design decisions that affect software architecture [7]. This paper focused on VT to represent SA, not concerning about the evolution of SA and its understanding.

Telea, Voinea and Sassenburg (2010) [10] performed a survey regarding the use of visual tools for SA understanding from the perspective of stakeholders. They analyzed the results using software architecture visualizations tools (AVTs) aiming to orientate industrial practitioners in the adoption of tools and techniques according requirements and capabilities of each type. For this survey, they defined three stakeholder types: a technical users (developers), project managers/lead architects, and consultants. They concluded that the AVTs were more adequate for technical users and less adequate for consultants, according to expectancy of each stakeholder [10]. Although this paper explored the visual solutions for SA understanding, it did not focus on SA evolution solutions.

Breivold, Crnkovic and Larson (2012) [1] conducted a systematic review focusing on software architecture evolution. The goal of the review was to get an overview at the architectural level about existing approaches in the analysis and improvement of software evolution, and also examine the impacts on research and industry. The authors identified five main categories of themes: (i) techniques supporting quality consideration during software architecture design, (ii) architectural quality evaluation, (iii) economic valuation, (iv) architectural knowledge management, and (v) modeling techniques [1]. The conclusion of this study emphasized the need of development of methods, process and/or tools to design architecture in a very large systems involving many platforms, organizations and processes, considering the amount and particularity of artifacts produced and used during its lifecycle. This study also presented several useful conclusions for researchers and practitioners, including considering the possibility to elaborate new ideas beyond Lehman's lays (about software evolution) so long as it has practical value for evolution. Additionally, this paper also reported the existence of few works for this area and expected that happens more research works in the future. [1]. Despite this article has studied the evolution of SA, it did not emphasize on visual solutions to support the comprehension of SA evolution.

Despite the large contribution of the aforementioned related works to the SA area, including SA evolution, they do not focus on visual solutions to support the comprehension of the SA evolution.

3. Research Design

We conducted a SMS to find evidence for the use of visual solutions to support the comprehension of SA evolution during the software lifespan. A SMS is a form of a systematic literature review (SLR) with more general research questions, aiming to provide an overview of the given research [11]. We decided to conduct a SMS due to the potential that this methodology has to reduce the analysis bias, through the establishment of selection procedures [12].

3.1. Planning

We conducted this SMS based on a protocol comprised of objectives of the review, criteria for considering papers, research questions, selected electronic databases and its search strings, selection procedures and exclusion, inclusion and quality criteria to select the studies from which we aim to answer the stated research questions [12]. The protocol of this SMS and related artifacts are available in a public Github repository [1]. The goal of this study is presented in Table 1 according to the GQM approach [13].

Table 1: The Goal of this SMS according to the GQM Approach

Analyze	the use of visual solutions
for the purpose of	comprehension
with respect to	software architecture evolution
from the point of view of	software architects and developers
in the context of	both academia and industry

The Research Question (RQ) is "*How have researchers and practitioners from the industry considered the use of software visual solutions to support the comprehension of software architecture evolution based on papers published in the peer-reviewed literature?*". This research question is in line with the goal of this review, and has been derived into four specific research questions, as follows: Specific Research Question 1 (SRQ1): *What are the main visual solutions to support the software architecture evolution comprehension?* Specific Research Question 2 (SRQ2): *What are the purposes of each visual solution to support the software architecture evolution comprehension?* Specific Research Question 3 (SRQ3): *How the solutions designed to visually support comprehension of software architecture evolution can be classified?* Specific Research Question 4 (SRQ4):

[1] https://github.com/dmsviva19saevolution/submission

Which visual forms are used to support comprehension of software architecture evolution?

The motivation behind RQ is justified by the acknowledgment that the comprehension of the software architecture evolution is required to tackle issues or improvements related to the software architecture and its evolution throughout releases [14] [5] [8] [7] [6] [2]. The specific research questions have the goal to gather evidence to support the answer of the stated RQ.

We considered the PICO criteria to define the search strings, as shown in Table 2. The search strings are based on this criteria for the selective process of papers for this review.

Table 2: PICO Criteria for Search Strings

(P)opulation	studies of software architecture in software engineering
(I)ntervention	visual solutions to support the comprehension of software architecture evolution
(C)omparison	not applicable
(O)utcomes	solutions (i.e., tools, techniques, environment, approaches, models or methodology) with focus on visual resources to support software architecture evolution; visual solutions to software architecture evolution; use of visual resources to comprehension of software architecture evolution

The formation of the search string applied in the electronic databases is shown in Tables 3 and 4. The Table 3 refers to major terms for the research objectives, built using the PICO criteria. We also considered the use of alternative terms and synonyms of these major terms. For example, the term *visualization* can be associated with terms such as *visual*, *visualizing* and *visualize*. These alternative terms, as shown in Table 4,are also included in the search string. We built the final search string by joining the major terms with the boolean "AND" and joining the alternative terms to the main terms with the boolean "OR". The focus of the formed search strings is to focus on papers targeting the research questions of this systematic mapping.

Table 3: Major terms for the research objectives

Criteria	Major Terms
(P)opulation	AND "software engineering" AND "software architecture"
(I)ntervention	AND "comprehension" AND "evolution"
(C)omparison	Not Applicable
(O)utcomes	AND "visual" AND "solution"

Table 5 presents the electronic databases from which we retrieved the papers along with the respective search strings used to retrieve the papers. Table 6 presents the criteria for exclusion and inclusion of papers in this review. The *OR* connective adopted in the exclusion criteria, means that the exclusion criteria are independent, i.e., meeting only one criterion is enough to exclude the paper. On the other hand, the *AND* connective in the inclusion criteria, means that all

Table 4: Alternative terms from majors terms

Major Term	Alternative Terms
"evolution"	("evolution" OR "evolve" OR "evolving")
"comprehension"	("comprehension" OR "understanding" OR "understand" OR "support" OR "analysis" OR "evaluation" OR "examination" OR "explore" OR "exploring")
"solution"	("tool" OR "environment" OR "technique" OR "approach" OR "model" OR "methodology" OR "solution")
"visual"	("visualization" OR "visualizing" OR "visualize" OR "visual")

Table 5: Electronic Databases Selected for this SMS

Database and URL	Search Strings
Scopus www.scopus.com	("software architecture" AND ("evolution" OR "evolve" OR "evolving") AND ("comprehension" OR "understanding" OR "understand" OR "support" OR "analysis" OR "evaluation" OR "examination" OR "explore" OR "exploring") AND ("tool" OR "environment" OR "technique" OR "approach" OR "model" OR "methodology" OR "solution") AND ("visualization" OR "visualizing" OR "visualize" OR "visual") AND "software engineering")
ACM Digital Library portal.acm.org	(+"software architecture" +("evolution" "evolve" "evolving") +("comprehension" "understanding" "understand" "support" "analysis" "evaluation" "examination" "explore" "exploring") +("tool" "environment" "technique" "approach" "model" "methodology" "solution") +("visualization" "visualizing" "visualize" "visual") +"software engineering")
Engineering Village (Ei Compendex) www.engineeringvillage.com	("software architecture" AND ("evolution" OR "evolve" OR "evolving") AND ("comprehension" OR "understanding" OR "understand" OR "support" OR "analysis" OR "evaluation" OR "examination" OR "explore" OR "exploring") AND ("tool" OR "environment" OR "technique" OR "approach" OR "model" OR "methodology" OR "solution") AND ("visualization" OR "visualizing" OR "visualize" OR "visual") AND "software engineering")
IEEE Xplore ieeexplore.ieee.org	("software architecture" AND ("evolution" OR "evolve" OR "evolving") AND ("comprehension" OR "understanding" OR "understand" OR "support" OR "analysis" OR "evaluation" OR "examination" OR "explore" OR "exploring") AND ("tool" OR "environment" OR "technique" OR "approach" OR "model" OR "methodology" OR "solution") AND ("visualization" OR "visualizing" OR "visualize" OR "visual") AND "software engineering")

inclusion criteria must met to select the paper under analysis. Table 6 also presents the quality criteria used for this review represented as questions that were adopted and adjusted from Dyba and Dingsoyr [15]. A critical examination following the quality criteria established in this table was performed in all remaining papers that passed the exclusion and inclusion criteria. All these criteria must met (i.e., the answer must be YES for each one) to permanently select the paper, otherwise the paper must be excluded. The exclusion, inclusion and quality criteria were used in the selection process as presented in Table 7. According to Table 8, at the end of the selection process, all the retrieved papers were classified in one of the three options: *Excluded*, *Not Selected* and *Selected*.

3.2. Execution

The quantitative evolution of papers throughout the execution of this SMS is summarized in Figure 1. The figure uses the PRISMA flow diagram [16] and shows the performed steps and the respective number of documents for each phase of the SMS, following the outline described in Subsection 3.1.

Table 9 presents the effectiveness of the the search strings showed in Table 5 considering the 92 retrieved pa-

Table 6: Exclusion, Inclusion and Quality Criteria

Type	Id	Description	Connective or Answer
Exclusion	E1	Published earlier than 2000	OR
Exclusion	E2	The paper was not published in a peer-reviewed journal or conference	OR
Exclusion	E3	The paper does not present a primary study	OR
Exclusion	E4	The paper is not written in English	OR
Exclusion	E5	The paper has less than 3 pages	OR
Inclusion	I1	The paper must present an approach in the usage of visual solution to support the comprehension of software architecture evolution	AND
Quality	Q1	Are the aims of the study clearly specified?	YES/NO
Quality	Q2	Is the context of the study clearly stated?	YES/NO
Quality	Q3	Does the research design support the aims of the study?	YES/NO
Quality	Q4	Has the study an adequate description of the visual solution?	YES/NO
Quality	Q5	Is there a clear statement of findings by applying the visual solution to support the comprehension of software architecture evolution?	YES/NO

Table 7: Steps of the Selection Process

Step	Description
1	Apply the search strings to obtain a list of candidate papers in specific eletronic databases.
2	Remove replicated papers from the list.
3	Apply the exclusion criteria in the listed papers.
4	Apply the inclusion criteria after reading abstracts, introduction and conclusion in papers not excluded in step 3.
5	Apply quality criteria in selected papers in step 3.

Table 8: Classification Options for Each Retrieved Paper

Classification	Description
Excluded	Papers met the exclusion criteria.
Not Selected	Papers not excluded due to the exclusion criteria, but did not meet the inclusion or quality criteria.
Selected	Papers did not meet the exclusion criteria and met both the inclusion and quality criteria.

pers. The electronic database that more contributed with selected studies was the *IEEE Xplore* with five papers, corresponding to a search effectiveness of 18.5%. The twelve selected papers represented 13.0% of all 92 retrieved papers.

The Figure 2 presents a overview of contribution of each exclusion criterion in total of excluded papers. The exclusion criterion that had more contribution was the E1 criterion that says "*Published date less than 2000*", accounting for 54% of excluded papers. Note that the E4 criterion ("*The paper is not written in english*") did not have any occurrence.

The Figure 3 presents graphics that provide a overview of sources (electronic databases) distribution by papers status. The papers status is according to Table 8. The Figure

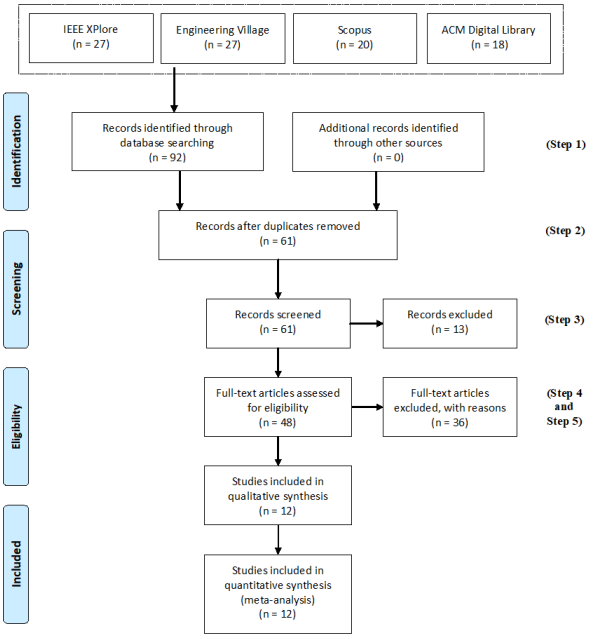

Figure 1: Procedures and its results in the papers selection process.

Table 9: Effectiveness of the Search Strings

Database	Selected Papers	Excluded Papers	Not Selected Papers	Replicated Papers	Total Search Results	Search Effectiveness
ACM Digital Library	2	2	13	1	18	11.1%
Engineering Village	3	4	6	14	27	11.1%
IEEE Xplore	5	4	14	4	27	18.5%
Scopus	2	3	3	12	20	10.0%
TOTAL	**12**	**13**	**36**	**31**	**92**	**13.0%**

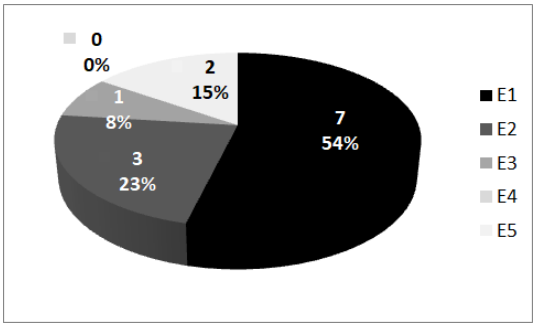

Figure 2: Contribution of exclusion criteria in total of excluded papers.

3a shows the selected papers distribution by source. The "IEEE Xplore" had the major contribution for selected papers with 41% of occurrences. The "Engineering Village"

was the second with 25% of selected studies. The Figure 3b shows the not-selected papers distribution by source. The "IEEE Xplore" leaded with 39% of not-selected papers, close to "ACM Digital Library" with 36%. The Figure 3c presents the excluded papers distribution by source. The "Engineering Village" and "IEEE Xplore" databases had the majors contributions with 31% of excluded papers each one. The Figure 3d presents the distribution of all found papers by source. The "Engineering Village" and "IEEE Xplore" databases had the majors contributions, both with 29% of all papers found in its databases searches.

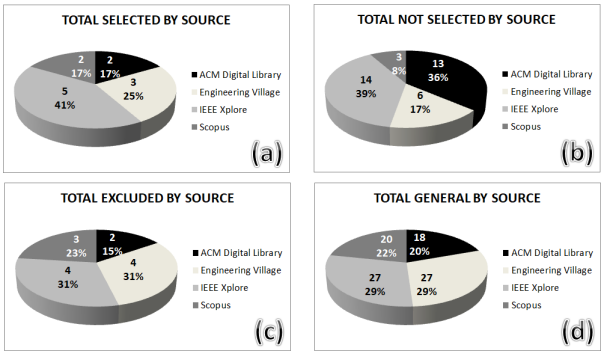

Figure 3: Overview of paper totalization by status and by electronic databases.

4. Results

The Table 10 shows the list of 12 selected papers of this systematic mapping. All papers are labeled as "S" followed by the paper reference number through which the paper can be reached at the end of this document. All the selected papers were published in conferences. Unfortunately, the retrieved papers from journals matched the exclusion criteria (Step 3 of Table 7) or did not matched the inclusion criteria (Step 4 of Table 7). For this reason, they were discarded from this systematic mapping.

The paper **S01** [3] describes a solution aimed at enhancing the comprehension of the software architectural evolution based on visual resources. This solution proposes the use of efficient navigation and visualization of the history of software architectural changes throughout releases, integrating the use of evolution metrics with software visualization techniques. This integration has the goal to support both tracking and analysis of architectural changes from past releases. The authors developed the so called *Origin Analysis* method to analyze software structural change. This method supports the identification of possible origin of function or file that appears to be new in a later release of the software system, if it already existed in the system elsewhere [3]. This method highlights the use of two techniques in its implementation: *Bertillonage Analysis* and *Dependency Analysis*. The paper also performs a study of evolution of a real tool to demonstrate the use of *BEAGLE*, a prototype implementation of this solution that works as an integrated environment for studying software architecture evolution, as a validation form of the *Origin Analysis* [3]. This paper does not discuss explicitly its limitations, even though cites some of them.

Table 10: Summary List of Selected Papers

Ref. Label	Title	Conference	Year
S01	An integrated approach for studying architectural evolution [3]	IEEE Workshop on Program Comprehension	2002
S02	An Approach based on Bigraphical Reactive Systems to Check Architectural Instance Conforming to its Style [17]	First Joint IEEE/IFIP Symp. Theoretical Aspects of Software Engineering (TASE)	2007
S03	Exploring inter-module relationships in evolving software systems [4]	European Conference on Software Maintenance and Reengineering (CSMR)	2007
S04	The SAVE Tool and Process Applied to Ground Software Development at JHU/APL: An Experience Report on Technology Infusion [18]	IEEE Software Engineering Workshop (SEW)	2007
S05	Development of a methodology, software-suite and service for supporting software architecture reconstruction [19]	European Conference on Software Maintenance and Reengineering (CSMR)	2010
S06	Evolve: tool support for architecture evolution [20]	International Conference on Software Engineering (ICSE)	2011
S07	Model-Based Software Architecture Evolution and Evaluation [21]	Asia-Pacific Software Engineering Conference (APSEC)	2012
S08	ECITY: A tool to track software structural changes using an evolving city [22]	International Conference on Software Maintenance (ICSM)	2013
S09	Run-time monitoring and real-time visualization of software architectures [23]	Asia-Pacific Software Engineering Conference (APSEC)	2013
S10	eCITY+: A Tool to Analyze Software Architectural Relations Through Interactive Visual Support [24]	European Conference on Software Architecture Workshops	2014
S11	Towards the understanding and evolution of monolithic applications as microservices [25]	Latin American Computing Conference (CLEI)	2017
S12	EVA: A Tool for Visualizing Software Architectural Evolution [2]	International Conference on Software Engineering (ICSE)	2018

The paper **S02** [17] proposes a graphical technical description of the architectural instance of a software system, and verify the compliance to its corresponding style. This solution is based on the *Bigraphical Reactive Systems* (BRS) to perform the verification with formal methods, and uses an extended version of a Bigraph to describe the instance. Besides supplying a visual method to specify ar-

chitectural instances and styles, the solution proposed can enhance the ability to design evolving systems. Additionally, the paper shows two study cases in order to prove the effectiveness of this solution [17].

The paper **S03** [4] proposes an approach based on the visual representation of inter-module dependencies and relationships between SA components and modules throughout multiple versions of the software system. The *Semantic Dependency Matrix* is a visualization technique that shows dependencies between two modules with similar behavior classes. The *Edge Evolution Film Strip* is a visualization technique that presents the evolution of an inter-module relations in a software system along its multiples versions. These techniques were applied in two large open source software systems, in reverse engineering context, to exemplify them. The paper also purposes a pattern language for inter-module relationships. The studied examples are provided from an exploration prototype named *Softwarenaut* [4].

The paper **S04** [18] describes the NASA JHU/APL's experiences in using the *SAVE (Software Architecture Visualization and Evaluation*) tool and process. The *SAVE* tool addresses the understanding, maintenance and evolving issues, allowing software architects to navigate, visualize, analyze, compare, evaluate, and improve their software systems, all in only one environment. This tool can be also used to develop a new architecture, compare with the current one and still helps in change impact analysis, among others features. The architecture comparison can also occur between distinct software systems. The paper shows how the *SAVE* tool has been successfully applied to the Common Ground software, a shared software architecture used by NASA missions software systems, in order to avoid further SA maintenance and evolution problems [18]. However, this paper does not discuss the limitations of its study.

The paper **S05** [19] presents the description and goals of the project titled *"Development of a methodology, software-suite and service for supporting software architecture reconstruction"*, intended to develop a methodology and a tool-set (environment) to do automatic architecture reconstruction of software systems through visual resources utilization. It also provides tracking of changes in architectural components during software evolution. At the time this paper was written (2010) the project was focused to systems that has been built using Java or .NET technologies and deal with SQL databases. This paper presents limitation of its study, but only in a summarized way. It also shows details of the current status (2010) of this project [19].

The paper **S06** [20] introduces *Evolve*, a graphical modeling tool that implements an ADL (*Architectural Description Language*) named *Backbone* that is focused on software architecture evolution. *Evolve* supports definition and evolution of SA using the *Backbone*, with particular attention to incremental change and unplanned change processing, very common activities in the software development and evolution process. *Backbone* provides constructs that allow changes that may result in architectural anomalies but *Evolve* is able to detect these anomalies. The paper shows the main characteristics of *Evolve* and how it deal with changes definition in SA, besides a brief use historic. The *Evolve* tool, at the time the paper was published (2011), was freely available for academic research and the production of open source software under the *GNU Affero General Public License version 3*.

The paper **S07** [21] presents and proposes the development of *ARAMIS (Architecture Analysis and Monitoring Infrastructure)*, an architecture meta-model based solution to extract run-time architecture information and provide data to generate new dynamic architecture views in real time. The solution provides visual representations of the monitored architecture at several abstraction levels, as well as the availability of methods to evaluate this architecture [21]. This study does not present any type of implementation (only technical description), despite presents some limitations of the solution.

The paper **S08** [22] introduces *eCITY*, a tool that helps software architects and developers to understand the software structure of their system. It allows the track of components' insertion, removal, or modification over system lifespan and provides an interactive visualization that provides an overview of changes. All of this implemented under a city metaphor using animations to represent the transitions of the architecture components and color coding to highlight the evolution and changes of these components (Figure 4). The *eCITY* provides an overview of the entire system at a desired point into its evolution process (life cycle), implementing it under a city metaphor, allowing the user an interactive way to understand and explore these changes. *eCITY* provides views to help the changes over time, like: *Timeline View*, an administrative view that uses charts and color to emphasize changes between software system versions; and *City View*, a city layout using animations to represent the transitions of the architecture components and color coding to highlight the evolution and changes of these components, as shown in Figure 4. The *eCITY* tool works with compile-time information, not providing dynamic views [22]. The *eCITY* was originally designed as an Eclipse plug-in. This paper also presents a summary of a conducted user study to emphasize its usefulness.

The paper **S09** [23] describes the implementation results of some core characteristics of *ARAMIS*, previously proposed in S07 [21]. *ARAMIS* is a approach for evolution and evaluation of software systems that relies on a infrastructure of run-time monitoring to manage the behavior of the system, in several abstraction levels. In this paper, a prototype of *ARAMIS* was developed focused only in reconstruc-

Figure 4: *eCITY*: a City View [22]

tion of object-level interactions. The prototype uses aspect-oriented techniques to extract and gather the run-time architectural information, and the *XMPP (Extensible Messaging and Presence Protocol)* to distribute the gathered information for visualization in real-time, through specialized components, as we can see in Figure 5. The evaluation of this process, according to prototype results, shows that *ARAMIS* can easily be used to demonstrate the behavior of the run-time monitored systems [23].

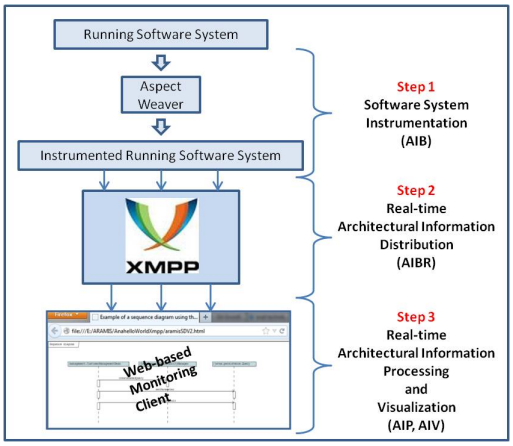

Figure 5: *ARAMIS*: a Prototype Overview [23]

The paper **S10** [24] presents *eCITY+*, an improved version of eCITY, presented in S08 [22], now combining the stable city layout and the *Hierarchical Edge Bundling (HEB)*, an useful technique to help the implementation of 3D visualization with the use of animation. The *eCITY+* is a later version of *eCITY* tool presented in S08 [22], owning characteristics similar from its predecessor. The *eCITY+* tool primarily differs from its earlier version in the use of *HEB* to highlight changes in both the hierarchical structure as well as the inter-dependencies between software components. The *eCITY+* tool, as its predecessor, performs analysis of software architecture relationships through interactive visual support, using the city metaphor to provide an overview of the entire system [24]. The *eCITY+* also was consciously developed as a plug-in to traditional SA maintenance tools.

The paper **S11** [25] describes a technical solution to modernize monolithic applications into microservices using software visualization to support the comprehension of evolution process. This conceptual solution can provide a modernization process that uses a legacy system and generates a set of visual diagrams that help architects and developers to understand the system, also suggesting ways of code partitions for transforming into micro-services. The paper has focus only in an understanding stage of modernization, not in transformation stage [25]. The authors analyzed a large Java EE application to validate this solution.

The paper **S12** [2] introduces *EVA (Evolution Visualization for Architectures)*, a visual tool to help software architects understand the evolution of architecture and therefore track and analyze architectural changes. This tool can visualize and explore architectures of software systems with a long life cycle, including stages of its evolution. *EVA* provides three main views: *Single-Release Architecture View*, that shows the architecture of only one software system version, as shown in Figure 6a; *3-D Architecture-Evolution View*, that depicts architectures of multiple software system versions in a single compositional view, as shown in Figure 6b; and Pairwise Architecture-Comparison View, that presents the architectural differences between two software system versions, as shown in Figure 6c. *EVA* allows its users to assess the impact of design decisions, as well as its rationale, which have influenced in the software architecture. As we can see in Figure 6, *EVA* uses color coding to distinct packages or groups of code level entities. The work is currently (2018) focused on showing the explicit reasons behind the architectural changes, in order to assist the rationale of tracking design during the software lifespan [2]. This article presents limitations of its study, but does not discuss them explicitly. The *EVA* tool was developed in Phyton and is available in a GitHub repository.

Unfortunately, we could not find any taxonomy to classify solutions that support the comprehension of SA evolution using visual resources. Then, to improve the understanding of these visual solutions previously described, we propose a taxonomy with the goal to classify their main characteristics, properties and features. Next, we present the taxonomy and the corresponding classification of each of the selected visual solutions in this SMS.

4.1. Taxonomy for Visual Solutions to Support SA Evolution Comprehension

In this subsection, we present the resulting taxonomy of visual solutions to support the comprehension of software architecture evolution. We argue that the concepts presented in this taxonomy are key to understand the char-

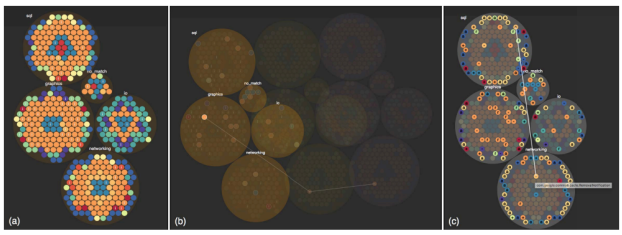

Figure 6: Three Types of Visualization of *EVA* [2]

acteristics of visual solutions and therefore to answer the research questions of this secondary study. We used the software visualization taxonomy proposed by Price, Baecker and Small (1993) [26] and the framework purposed by Gallagher, Hatch and Munro (2008) [27] as references. The first focused on software visualization, whereas the latter focused on software architecture. For this reason, we adopted them as references to propose the new taxonomy. According to Sulr et al. [11], a taxonomy is consisted of a number of dimensions (e.g., visualization form) with their attributes (e.g., 2D Elements, Color Coding). Each visual solution from the selected papers from this SMS can pertain to one or more attributes from a specific dimension, as we will describe in the following paragraphs.

As can be seen in Table 11, the **Category** dimension is related to the type of proposed visual solution to support the comprehension of SA evolution. It is comprised of four attributes: *Description*, *Technique*, *Tool* and *Environment*. The *Description* attribute classifies the solution or its core idea as a main concept or a technical description of a visual solution. The *Technique* attribute informs whether the visual solution proposes the use of a technique, such as specialized procedures or processes. The *Tool* attribute means the solution presents or proposes a tool , or its development, to assist specialized users of software architecture to develop, or maintain software systems. The *Environment* attribute indicates that the solution explicitly presents or proposes an environment, i.e., an integrated set of tools to help software architecture specialized users in the development or maintenance of software systems. The value of this dimension for each solution is mandatory and it may have more than one attribute signalized.

The **Stage** dimension has four attributes. The *Conceptual* attribute indicates that the solution is still represented and referenced as a *concept*, not having any implementation. The *Project* attribute indicates that the solution has been under development as a project. The *Prototype* attribute means the solution is a prototype of the solution. Finally, the *Stable Release* attribute is related to the status of the solution as already in production as a stable release. The value of this dimension is also mandatory and only one attribute can be signalized for each solution.

The **Visualization Form** dimension defines the visual characteristics of the output of the solution [26]. It is characterizes by eight attributes as follows. The *2D Elements* attribute indicates that the solution uses 2D elements, such as 2D charts, diagrams, shapes, windows, figures and lines. The *3D Visualization* attribute indicates that the solution uses 3D resources for visualization. The *Animation* attribute means the solution uses resources of animation in the visual representations. The *Bigraphs* attribute marks the use of bigraphs in the visual solution. The *Visual Metaphor* attribute indicates the use of one or more visual metaphors in the visual representations to enhance the SA evolution understanding. The *Color Coding* attribute means the solution adopts a specific color system to represent the data. The *Tree-based* attribute is used to characterize solutions that use visual structures based on trees. The *UML-based* indicates that the solution uses UML diagrams as a form of visual representation.

The **Static Representation** dimension shows what architectural information can be extracted and represented before run-time [27]. It has two associated attributes: *Static Visualization* and *Recovery*. The *Static Visualization* attribute means the solution displays data exclusively related to static structure of the software system. The *Recovery* attribute indicates that the solution supports the retrieval of architectural data from specific sources. This dimension may have no attributes flagged.

The **Dynamic Representation** dimension shows what architectural information can be extracted and represented during run-time [27]. The *Dynamic Visualization* attribute means the solution displays data extracted during its execution (run-time). The *Events Monitoring* indicates if the solution perform the catch events during its execution. These events can be identified and associated with SA elements and thereby support the comprehension of specific scenarios of software architecture evolution. The *Live* attribute points out that the SA data is gathered in a *real time* fashion as the solution is executed [26] [27]. The *Post-mortem* attribute means the SA data to be gathered is produced in a *post-mortem* fashion by the solution, i.e., generated by its previous execution [26] [27]. This dimension may have no attributes flagged.

The **Architectural Tasks** dimension is related with features of the visual solution that support stakeholders to perform tasks that to some extent focuses on the software architecture and its evolution [27]. It has nine attributes as follows. The *Anomalies* attribute indicates that the solution supports the identification of anomalies, violations and inconsistencies occurrences related to SA. These occurrences can also influence the *Comprehension* attribute indicates the solution supports visual analysis tasks to improve the comprehension of SA and its evolution. Analysis tasks means tasks that generate results to facilitate the understanding the SA, its components, dependencies and relationships, as well

as its evolution. They should support top-down or bottom-up approaches [27]. The *Styles* attribute indicates that the solution is able to identify architectural styles and/or verify its compliance with a predefined reference. The *Show Evolution* attribute indicates that the solution provides facilities to exhibit evolution evidence of a SA, in a basic or advanced way [27]. The *Construction* attribute indicates that the solution provides resources to add, change or remove SA elements in the visual representation. The *Evaluation* attribute means the solution supports SA quality analysis and also compliance evaluation. The *Comparison* attribute points out that the solution performs visual comparison among releases of the software system under analysis. A typical use of this attribute is the comparison between the *as-is* with the *to-be* architectures or the *as-designed* with *as-implemented* software architecture [27]. The *Tracking* indicates that the the solution supports the tracking of SA changes throughout its releases. This is a key resource to, for example, identify and trace the architectural decay of a SA, which impairs the software lifespan [2]. Finally, the *Rationale* indicates that the solution presents and make available the rationale behind the design decisions that somehow influences the SA.

4.2. Visual Solutions According to Taxonomy

The Table 11 shows the main characteristics, properties and features identified in the visual solutions of the selected papers from the perspective of the proposed taxonomy. These characteristics were identified and collected exclusively based on the text provided by the selected studies listed in Table 10.

The column labeled *Category* indicates different categories of solutions found in the selected papers, according the description presented in Subsection 4.1. The column labeled *Stage* means the stage of the solution proposed by the paper at the time it was published. The column labeled *Visualization Form* means the summary description of fundamental characteristics related to what can be exhibited in the visual solution. The content of this column is based on the *Form* category proposed by the taxonomy of Price [26]. The columns labeled *Static Representation*, *Dynamic Representation*, *Architectural Tasks* are based on keys areas proposed in Gallagher's framework [27]. The column *Others Features* shows complementary purposes, features and characteristics of the presented visual solution not listed before.

We decided to present a analysis about the visual solution named *EVA* to illustrate how the characteristics of the selected visual solutions presented in Table 11 can be determined. This analysis is shown in Table 12. The visual solution *EVA* was previously presented in the paper S12 [2] and its justifications come exclusively from this paper's content.

The Figure 7 presents the overview of current stages for all visual solutions referenced by Table 11. Its worth remembering that current stages means the stage at the time its study was published. Note that most of solutions (5 in 12) were in "Stable Release" stage, whereas few of them (2 in 12) were in "Conceptual" stage.

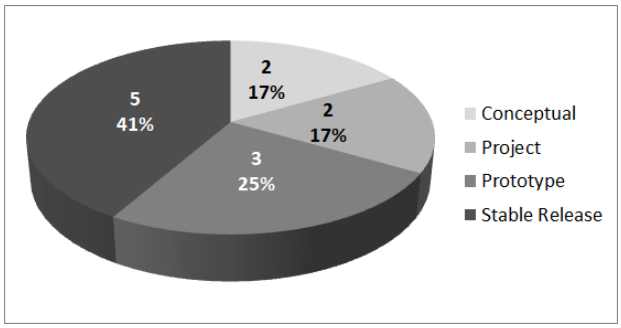

Figure 7: Current Stages of Visual Solutions

The Figure 8 shows the categories of solution found in selected papers. The "Tool" category has the major preference in visual solutions, being present in 8 of 12 solutions found. The "Description" category is present in only two solutions.

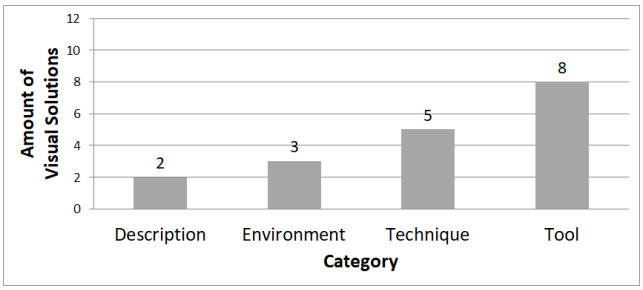

Figure 8: Categories of Visual Solutions

Figure 9 shows that the tasks *Comprehension* and *Show Evolution* are adopted in all visual solutions from the selected studies. It means that all of them reported the support of analysis tasks to improve the SA comprehension. Moreover, they also reported the adoption of facilities to exhibit the SA evolution. These are in fact, minimum requirements of a visual solutions to support comprehension of SA evolution. The task *Comparison* is also representative in the analyzed solutions. They have reported the ability to perform visual comparison of SA characteristics among two or more releases of a specific software system, which corresponds to 83% of analyzed visual solutions. The visual solutions not related to the task *Comparison* is the ARAMIS solution (S07) [21] and (S09) [23]. These papers reported the focus in real-time monitoring. The tasks *Anomalies*, *Styles* and *Rationale* were not representative in the solutions, corresponding to only 17% . The task *Construction* explicitly appears in 33% of the visual solutions and half of them

Table 11: Using the Proposed Taxonomy to Classify the Visual Solutions to SA Evolution

Ref. Paper	Name of Visual Solution	Category	Stage	Visualization Form	Static Representation	Dynamic Representation	Architectural Tasks	Other Features
S01	Beagle	Environment, Technique	Prototype	2D Elements, Tree-based	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution	Evolution metrics usage
S02	Not named	Description, Technique	Conceptual	2D Elements, Bigraph	Static Visualization	Dynamic Visualization, Live	Comprehension, Comparison, Construction, Show Evolution, Styles	BRS resources
S03	Film Strip and Dependency Matrix	Technique	Prototype	2D Elements	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution	N/A
S04	SAVE	Tool, Environment	Stable Release	2D Elements	Static Visualization, Recovery	N/A	Comprehension, Anomalies, Comparison, Construction, Show Evolution, Rationale, Styles, Evaluation	Process-oriented, Products comparison
S05	GOP	Tool, Environment	Project	2D Elements, Color Coding	Static Visualization, Recovery	N/A	Comprehension, Comparison, Construction, Show Evolution, Tracking	Methodology-oriented
S06	Evolve	Tool	Stable Release	2D Elements, UML-based, Animation	Static Visualization	N/A	Comprehension, Anomalies, Comparison, Construction, Show Evolution	Model-driven, ADL implementation
S07	ARAMIS	Tool	Conceptual	N/A	Static Visualization	Dynamic Visualization, Events Monitoring	Comprehension, Show Evolution, Evaluation	Model-driven
S08	eCITY	Tool	Stable Release	2D Elements, Color Coding, Animation, Visual Metaphor	Static Visualization	N/A	Comprehension, Comparison, Show Evolution, Tracking	N/A
S09	ARAMIS	Tool, Technique	Prototype	2D Elements, UML-based	N/A	Dynamic Visualization, Events Monitoring, Live, Post-mortem	Comprehension, Show Evolution, Evaluation	Model-driven, Traceability with requirements, Views creation
S10	eCITY+	Tool, Technique	Stable Release	2D Elements, 3D Visualization, Color Coding, Animation, Visual Metaphor	Static Visualization	N/A	Comprehension, Comparison, Show Evolution, Tracking	As-plugin
S11	Not named	Description	Project	2D Elements, UML-based, Color Coding	Static Visualization	N/A	Comprehension, Comparison, Show Evolution	Model-driven, Modernization
S12	EVA	Tool	Stable Release	2D Elements, 3D Visualization, Color Coding	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution, Tracking, Rationale	ADD Traceability

present specific resources to design the architecture, such as S02 solution [17] (using *Bigraphical Reactive System*) and S06 solution [20] (using an ADL visual modelling component).

5. Discussion

The specific research question SRQ1 is related to the main visual solutions to support the software architecture evolution comprehension. The answer is presented in Table 11, where it is possible to identify their main goals and characteristics of each solution. The specific research question SRQ2 is concerned to different purposes of using the visual solutions to support the software architecture evolution comprehension. The purposes are also presented in Table 11, identified primarily in the *Architectural Tasks* column and also in the *Static Representation*, *Dynamic Representation* and *Other Features* columns. The specific research question SRQ3 focuses on solutions designed to visually support comprehension of software architecture evolution can be classified. The solutions can be classified in the categories *Description*, *Technique*, *Tool* and *Environment*. The Table 11 classify each solution and shows that the *Tool* category has more representants among the selected studies. The specific research question SRQ4 focuses on visual forms used to support comprehension of software architecture evolution. The Table 11 uses the *Visualization Form* column to list the visual forms adopted in the visual solutions discussed in the selected studies.

Finally, the main research question (RQ) focuses on the evaluation of the usage of visual solutions to support the comprehension of software architecture evolution based on papers published in the peer-reviewed literature. Table 11 presents an up-to-date overview of solutions used for the stated purpose with different characteristics and strategies as has been already explained for the specific research questions. We have identified that based on evidence from the selected studies, features of comprehension and SA evolution visualization are minimum requirements to support software architecture evolution comprehension, as shown in Figure 9. On the other hand, the low number of papers found in this systematic mapping study, suggests that visual solutions to support SA evolution comprehension is an area

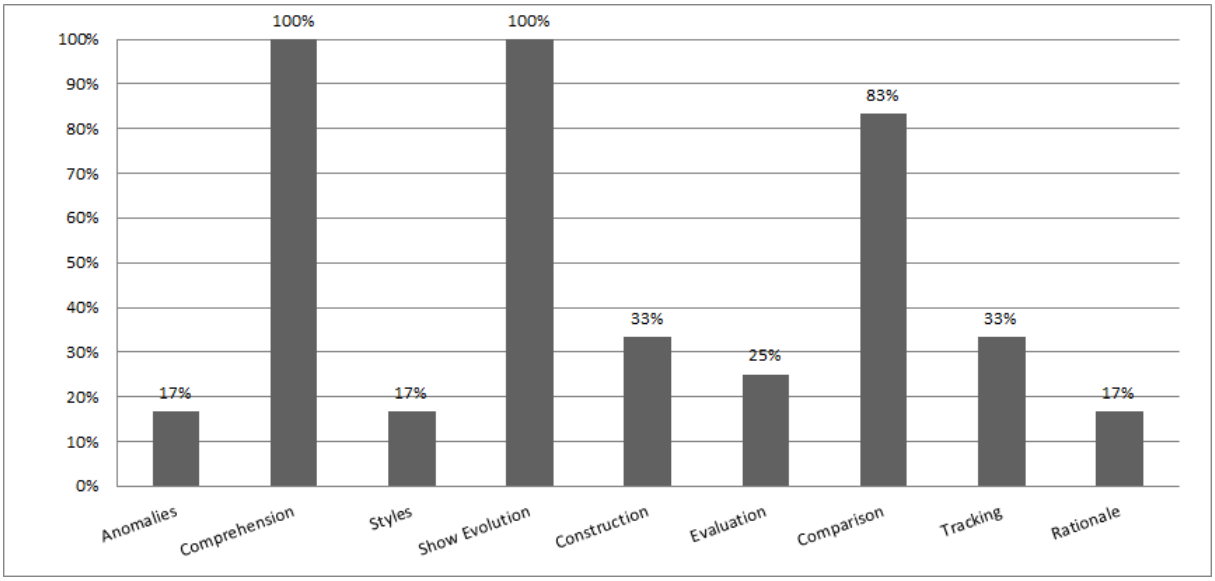

Figure 9: Distribution of Architectural Tasks in the Selected Papers

Table 12: Sample Analysis - Visual Solution *EVA* [2]

Values	**Justifications**
Category = ("Tool")	Title of the paper is *"EVA: A Tool for Visualizing Software Architectural Evolution"*.
Stage = ("Stable Released")	The behavior of *EVA* presented in the paper always suggests that it is in production; *EVA* was already evaluated in use and is in the process of deploying to a large development organization.
Visualization Form = ("2D Elements", "3D Visualization", "Color Coding")	"2D Elements": suggested by figures displayed in the paper; "3D Visualization": *EVA* provides a view that shows architectures of multiple releases in a 3D visualization; "Color Coding": EVA differs code-level entities through color coding.
Static Representation = ("Static Visualization", "Recovery")	"Static Visualization": *EVA* only collects data at compile-time (static elements); "Recovery": *EVA* supports many SA recovery techniques.
Dynamic Representation = N/A	The paper does not present any characteristic or feature for dynamic representation.
Architectural Tasks = ("Comprehension", "Comparison", "Show Evolution", "Tracking", "Rationale")	"Comprehension": *EVA* provides a view that represents a single SA release, allowing users to interactively understand the functionality of each architecture component; "Comparison": *EVA* provides comparison view that shows the SA dierences between two releases; "Show Evolution": *EVA* visualizes the SA evolution through a set of source codes across multiple releases; "Tracking": *EVA* provides a view that allows representation of change tracking of entities across multiple releases; "Rationale": *EVA* collects relevant data of design decisions from system repositories, displaying them together with the architecture evolution visualization.
Other Features = ("ADD Traceability")	*EVA* shows the traceability of architecture design decisions over time

that needs expand in terms of studies and options available for practitioners and researchers.

6. Conclusion

The aim of this work is to report the design, execution and results of a systematic mapping study of visual solutions to support the comprehension of software architecture evolution. We performed a SMS according to the plan described in Section 3. Initially, the applied search strings retrieved 92 papers from the selected electronic databases. This number was reduced to 12 after applying all selection procedures criteria. From these 12 selected studies, the identified visual solutions to support the comprehension of SA evolution were classified as follows: 17% categorized as *Description*, 42% as *Technique*, 67% as *Tool* and 25% as *Environment*. All of them support the architectural tasks *Comprehension* and *Show Evolution*.

Besides the identification of visual solutions from the literature, another contribution of this study is a taxonomy to classify these solutions. The taxonomy contains six dimensions: category, stage, visualization form, static representation, dynamic representation and architectural tasks. These dimensions and their attributes was explained in Section 4, SubSection 4.1. Each visual solution was classified according to this taxonomy, generating a table of characterization of visual solutions to SA evolution, presented in Table 11. The assignment of the taxonomy attributes value to the visual solution characteristics shows a current overview of the available visual solutions in peer-reviewed literature.

Moreover, this study also shows that visual solutions to support SA evolution comprehension usually present features to support analysis tasks to improve the SA compre-

hension and also provide facilities to exhibit the SA evolution. This study also concludes that, due to a few number of papers found in this SMS, the studies may consider to allocate more research and development effort to provide effective visual solutions to support SA evolution comprehension, improving the acquired knowledge in this area.

As a future work, we recommend extending the research to establish a methodology or process, based on the taxonomy proposed, to define projects to build visual solutions of SA evolution comprehension. Another possibility for future work is the improvement of the proposed taxonomy, aiming to generate a new framework to objectively evaluate solutions like that ones discussed in this SMS.

References

[1] H. P. Breivold, I. Crnkovic, and M. Larsson, "A systematic review of software architecture evolution research," Information and Software Technology, vol. 54, no. 1, pp. 16 – 40, 2012.

[2] D. Nam, Y. K. Lee, and N. Medvidovic, "Eva: A tool for visualizing software architectural evolution," in 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), May 2018, pp. 53–56.

[3] Q. Tu and M. W. Godfrey, "An integrated approach for studying architectural evolution," in Proceedings 10th International Workshop on Program Comprehension, 2002, pp. 127–136.

[4] M. Lungu and M. Lanza, "Exploring inter-module relationships in evolving software systems," in 11th European Conference on Software Maintenance and Reengineering (CSMR'07), 2007, pp. 91–102.

[5] R. Taylor, N. Medvidovic, and E. Dashofy, Software Architecture: Foundations, Theory and Practice, J. Wiley and Sons, Eds., Hoboken, New Jersey, 2009.

[6] D. Garlan, J. M. Barnes, B. Schmerl, and O. Celiku, "Evolution styles: Foundations and tool support for software architecture evolution," in 2009 Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture, 2009, pp. 131–140.

[7] M. Shahin, P. Liang, and M. A. Babar, "A systematic review of software architecture visualization techniques," Journal of Systems and Software, vol. 94, pp. 161 – 185, 2014.

[8] M. Shahin, P. Liang, and M. R. Khayyambashi, "Improving understandability of architecture design through visualization of architectural design decision," in Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge, ser. SHARK '10, 2010, pp. 88–95.

[9] R. L. Novais and M. G. de Mendonca Neto, Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications, ser. Chapter: Software Evolution Visualization: Status, Challenges, and Research Directions. IGI Global, 2018.

[10] A. Telea, L. Voinea, and H. Sassenburg, "Visual tools for software architecture understanding: A stakeholder perspective," IEEE Software, vol. 27, no. 6, pp. 46–53, 2010.

[11] M. Sulr, M. Bakov, S. Chodarev, and J. Porubn, "Visual augmentation of source code editors: A systematic mapping study," Journal of Visual Languages Computing, vol. 49, pp. 46 – 59, 2018.

[12] C. Wohlin et al., Experimentation in Software Engineering. Springer-Verlag, 2012.

[13] V. R. Basili and H. D. Rombach, "The tame project: towards improvement-oriented software environments," IEEE Transactions on Software Engineering, vol. 14, no. 6, pp. 758–773, 1988.

[14] J. Cleland-Huang, R. S. Hanmer, S. Supakkul, and M. Mirakhorli, "The twin peaks of requirements and architecture," IEEE Software, vol. 30, no. 2, pp. 24–29, 2013.

[15] T. Dyb and T. Dingsyr, "Empirical studies of agile software development: A systematic review," Information and Software Technology, vol. 50, no. 9, pp. 833 – 859, 2008.

[16] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, P. Group et al., "Preferred reporting items for systematic reviews and meta-analyses: the prisma statement," PLoS medicine, vol. 6, no. 7, p. e1000097, 2009.

[17] Z. Chang, X. Mao, and Z. Qi, "An approach based on bigraphical reactive systems to check architectural instance conforming to its style," in First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE '07), 2007, pp. 57–66.

[18] W. C. Stratton, D. E. Sibol, M. Lindvall, and P. Costa, "The save tool and process applied to ground software development at jhu/apl: An experience report on technology infusion," in 31st IEEE Software Engineering Workshop (SEW 2007), 2007, pp. 187–193.

[19] L. Schrettner, P. Hegedus, R. Ferenc, L. J. Fulop, and T. Bakota, "Development of a methodology, software – suite and service for supporting software architecture reconstruction," in 2010 14th European Conference on Software Maintenance and Reengineering, 2010, pp. 190–193.

[20] A. McVeigh, J. Kramer, and J. Magee, "Evolve: Tool support for architecture evolution," in Proceedings of the 33rd International Conference on Software Engineering, ser. ICSE '11, 2011, pp. 1040–1042.

[21] A. Dragomir and H. Lichter, "Model-based software architecture evolution and evaluation," in 2012 19th Asia-Pacific Software Engineering Conference, vol. 1, 2012, pp. 697–700.

[22] T. Khan, H. Barthel, A. Ebert, and P. Liggesmeyer, "ecity: A tool to track software structural changes using an evolving city," in 2013 IEEE International Conference on Software Maintenance, Sep. 2013, pp. 492–495.

[23] A. Dragomir and H. Lichter, "Run-time monitoring and real-time visualization of software architectures," in 2013 20th Asia-Pacific Software Engineering Conference (APSEC), vol. 1, 2013, pp. 396–403.

[24] T. Khan, S. R. Humayoun, K. Amrhein, H. Barthel, A. Ebert, and P. Liggesmeyer, "ecity+: A tool to analyze software architectural relations through interactive visual support," in Proceedings of the 2014 European Conference on Software Architecture Workshops, ser. ECSAW '14, 2014, pp. 36:1–36:4.

[25] D. Escobar, D. Crdenas, R. Amarillo, E. Castro, K. Garcs, C. Parra, and R. Casallas, "Towards the understanding and evolution of monolithic applications as microservices," in 2016 XLII Latin American Computing Conference (CLEI), 2016, pp. 1–11.

[26] B. A. Price, R. M. Baecker, and I. S. Small, "A principled taxonomy of software visualization," Journal of Visual Languages Computing, vol. 4, no. 3, pp. 211 – 266, 1993.

[27] K. Gallagher, A. Hatch, and M. Munro, "Software architecture visualization: An evaluation framework and its application," IEEE Transactions on Software Engineering, vol. 34, no. 2, pp. 260–270, 2008.