Developing a Conceptual Framework for Software Evolution Methods via Architectural Metrics

Nouredine Gasmallah, Abdelkrim Amirat, Mourad Oussalah, Hassina Seridi

▶ **To cite this version:**

Nouredine Gasmallah, Abdelkrim Amirat, Mourad Oussalah, Hassina Seridi. Developing a Conceptual Framework for Software Evolution Methods via Architectural Metrics. 6th IFIP International Conference on Computational Intelligence and Its Applications (CIIA), May 2018, Oran, Algeria. pp.140-149, 10.1007/978-3-319-89743-1_13 . hal-01913868

HAL Id: hal-01913868

https://inria.hal.science/hal-01913868v1

Submitted on 6 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Developing a Conceptual Framework for Software Evolution Methods via Architectural Metrics

N. Gasmallah[1,3], A. Amirat[1], M. Oussalah[2] and H. Seridi[3]

[1] Department of Mathematics and Computer Science
University of Souk-Ahras, 41000, Algeria
gasmallahedi@univ-soukahras.dz, a.amirat@univ-soukahras.dz
[2] Department of Computer Science
University of Nantes, 44300, France
mourad.oussalah@univ-nantes.fr
[3] Department of Computer Science
University of Annaba, 23000, Algeria
seridi@labged.net

Abstract. Because of the vital need for software systems to evolve and change over time in order to account for new requirements, software evolution at higher levels of modeling is considered as one of the main foundation within software engineering used to reduce complexity and ensure flexibility, usability and reliability. In similar studies for migration technique and software engineering, presenting a framework do not usually cover the specification of systems based on software architecture. In this paper, we specify a conceptual framework based on six explicit dimensions in respect to an architectural view-point as first class citizen. Indeed, sketching evolution relies upon identifying dimensions on which researchers try to answer while performing a new approach. The proposed model is based on answering What, Where, When, Who, Why and How questions. Analyzing these dimensions could provide a multiple choice to implement classification for architectural techniques. Further and using an example, these dimensions are quantified and then analyzed. This framework aims to provide a blueprint to guide researches to position architectural evolution approaches and maps them according a selected set of dimensions.

1 Introduction

Software evolution has become a major concern for stakeholders involved in the process of designing and modeling computer systems. Because of the rudimentary nature of software systems to evolve and change over time in order to account for new requirements and different needs, software evolution is considered as a vital pillar within the area of software engineering to ensure consistency and maintainability as well as to allow the system to open up for new directions and strategic opportunities. Further, software architecture must evolve within existing production systems as it constitutes the favorable blueprint for supporting

such evolvability at higher modeling levels [1]. This implies that evolution could be handled at earlier modeling phases where future changes could be anticipated. The area of software evolution has received unprecedented interest during the last two decades where numerous research studies have been published describing various methods and frameworks. In return, few studies have been devoted to present classification for software architecture evolution [2]. However, the main stream of most studies are either specialized in software architecture evolution as knowledge re-usability or addressing software evolution from an overall perspective [3, 4]. Buckley et al. (2005) proposed a taxonomy of software changes which characterize the mechanisms of change and their influencing factors. Williams et al. (2010) suggested that the key solution to address software changes is to identify the causes and effects which can be used to illustrate the potential impact of the change. The major drawback of their study is the lack of an explicit framework capable of positioning a given approach based on well-defined criteria or metrics. Introducing an evolution framework would fulfill the need to identify certain specifications according to which approaches can be classified. Because of the dearth of classification procedures that address the architectural evolution techniques with respect to well-defined dimensions, a conceptual framework that addresses the major concerns related to the evolution nature of software architecture is proposed in this research study. A framework can offer a structured and coherent design for examining the organizational aspect in software architecture evolution as well as assist architects to address decisions on new requirements. De facto, introducing a clear and concise architectural evolution framework would fulfill the need: i) to identify certain specifications according to the major concerns related to the evolution nature of software architecture and, ii) to provide a common vocabulary to understand, analyze and categorize a given evolution approach. The discussed model is inspired from the work of Buckley (2005) for software change in addition to the work of [5] for the architecture of information systems. The proposed framework is devised to reshape the dimensions reported in both earlier studies with a newly set of proposed metrics for the arena of software architecture evolution. Such metrics should assist to examine and assess evolution approaches through providing answers which revolve around these dimensions: What, Why, Where, Who, When and How the software architecture evolves? Based on these dimensions, the following research questions are addressed during this research study:

RQ1: What are the major metrics that can be used to analyze and compare the different architectural evolution methods?

RQ2 :Based on such dimensions, can a classification framework devises to regroup and categorize architectural evolution approaches?

The remainder of the paper is organized as follows: the proposed dimensions defined for positioning architectural evolution and a method for classifying them are explained in the following section. The third section illustrates the results drawn by the application of the proposed framework on a set of well-known frameworks in the field of software evolution. The penultimate section is dedi-

cated to discuss the findings to explore and compare the potentials of the proposed dimensions. Conclusions are drawn within the last section.

2 Proposed framework

2.1 Framework dimensions

What? Object of evolution dimension- The object of evolution is the subject on which the evolution is operated. It answers the what question and comprises of the following:

- **Artifact:** is an abstraction of any element belonging to the architectural structure. It may be a software architecture, a component, a service and so forth. It can be simple and even concrete description as program codes for example.
- **Process:** Process is as a set of interacting activities, which transforms inputs into outputs. Therefore the process also evolves and presents suitable means for anchoring the benefits pertaining to cost optimization and quality promotion [6], workflows in SOA are one of the best examples of processes.

Where? Hierarchical level dimension- Two types of hierarchical levels are commonly identified within the software engineering literature as depicted in Figure 1, which are modeling and abstraction levels.

- **Modeling levels ($\mathbf{M_0}$ to $\mathbf{M_2}$):** Software systems must be mapped over several levels, from lower-level constructs (code) to higher-level constructs, to ensure convergence [7]. Modeling level refers to one of the four levels (from M_0 to M_3) as defined by the OMG [8].
- **Abstraction levels ($\mathbf{a_0}$ to $\mathbf{a_n}$)**: Experts often use abstraction as an approach to address complex problems. This is still very often used for solving problems related to software modeling. Thereby, a solution is performed through a series of model description at a number of abstraction levels referred to in the Figure 1 as a_0, a_1..., a_n.

When? Time of evolution- In respect of architectural viewpoint, the time of evolution embraces one of these metrics:

- **Design-time:** Predicting evolution at the earlier stages of design allows improving and extending the system architecture. Model Driven Architecture is a good of methods addressing evolution at design-time, especially co-evolution approaches.
- **Run-time:** The evolution can occur while the system is running [9]. So far, evolving at run-time is considered as a major topic to address architecture adaptation [10]. Noting that, this metric encompasses: compile-time, Load-time and Dynamic-time [11, 10].

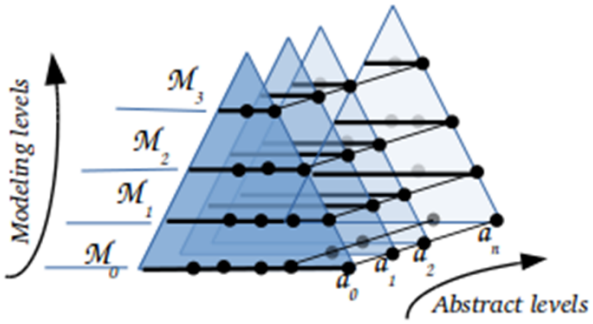

Fig. 1. Modeling Levels vs. Abstraction-Levels

Why? Type of evolution dimension- It is one of the most commonly-used metric when addressing classification issues. This dimension encloses the two following main sub-categories:
Main-categories: Instead of categories based on the architects intentions, main-categories sub-dimension is responding to the objective evidence of architect's activities identifiable from observations and artifacts before and after comparison of the software architecture.
Main-Forms of resolution: Architects can use several strategies for the evolution problem-solving according to different forms. From the literature, the following two major studies discussed the various forms in the area of evolution.
i) Chaki et al. [12] grouped the different resolution methods into two main strategies:

- *Open evolution* means from an initial architecture, of a new architecture reflecting a system solution in which a set of invariants and constraints are respected.
- *Close evolution* means the evolution activity uses induction resolution to find the best permissible sequence of operations to be applied to achieve the desired result.

ii) Oussalah et al.[13] proposed two other forms:

- *Break evolution* means that interventions are applied directly to the initial architecture without having the ability to go back on the trace of the evolution.
- *Seamless evolution* denotes an evolution with trace whereby the architecture keeps a trace of its initial properties and operations performed before each evolving process.

The proposed structuring for deducing the assumed evolution type for a particular method is shown in Table 1 such that the categorization is based hybridly on categories and forms of resolution. Herein, the proposed type of evolution is structured according to two main forms (curative or anticipative) which are explained as follows:

– **Curative-** when new requirements arise unpredictably during the life cycle.
– **Anticipative-** can be applied when evolution requirements are taken into account during the analysis.

Table 1. Structure of the type of evolution.

Type of Evolution	Main-forms				Main Categories		
	Open	Close	Break	Seamless	Corrective	Adaptive	Perfective
Curative	◯	•	⊙	◯	•	◯	◯
Anticipative	•	⊙	⊙	◯	◯	•	⊙

Symbol legend: • *higher* ⊙ *Medium* ◯ *Weak.*

The two main forms (*Curative* and *Anticipative*) can be set either to *Close* or *Open*. In the same way, the latter sub-forms can be set to *Break* or *Seamless*.

Who? Stakeholders- Stakeholders contribute in architecture enhancement in multiple roles regarding the responsibility they assume and the range of challenges they face [14]. For instance, this dimension covers only those stakeholders that operate evolution on the software system itself by referring to the development team enclosing *researches, architects, designers, developers, analysts* and *programmers* [10]. This dimension specifies two main knowledge about how are involved and interested by the evolution of software architecture.

How? Operating mechanism of evolution- The operating mechanism of evolution (OME) is introduced to refer to the general behavior and employed procedure by taking into account solely the described hierarchical levels. Mainly, evolution solving approaches commonly adhere to one of the two main following methods:

– **Reduce:** For the reductionist evolution, the process gets through a predefined evolution path; until the solution is satisfied (evolved model). Brooks [15]25 defines the reductionist approach as classical approach to problem solving to which the overall resolution task is decomposed into subtasks.
– **Emergence:** On the contrary during an emergentist evolution approach, the process builds the path to a solution. The emergence exposes a passage between the activity of micro-level and that of macro-level.

3 Case Study

3.1 Metric calculations

In order to clarify applicability of the proposed framework, two studies of architectural evolution are analyzed and compared. These studies are only cited

as an example for evolution approaches among many other relevant approaches. The first study by Oussalah et al. [16] presents a software architecture evolution model (SAEV) which aims to describe and manage the evolution through the architectural elements of a given software architecture. SAEV considers software architecture elements (component, connector, interface and configuration) as first class entities. Managing these elements is conceived independently of any architecture language level by considering the different levels of modeling (meta-level, architectural level and application level). The second by Barnes et al. [17] is an automatic approach provided to assist architects by planning alternative evolution paths for evolving software architecture. An evolution path is expressed in terms of intermediate architecture generated from the initial state. If the architect's goal is clear, this approach aims to assist architects to find the optimal path that meets the evolution requirements. The evolution path acts explicitly on software architecture and applies evolution from a higher to lower level to reach the desired architecture. This approach helps architects to find the optimal path for the evolved architecture. In order to apply the conceptual framework for the two approaches, all the discussed dimensions are quantified using three values 1, 0.5 and 0 which reflect respectively explicit, implicit and not mentioned. Conventionally, a dimension, which is explicitly mentioned, is assigned the value one. However, if it is shown in implicit fashion, 0.5 is assigned. Otherwise, the estimated value is zero. The values obtained from Table 2 shows that the two approaches have considered the six dimensions. For the where dimension for example, the first approach has explicitly expressed evolution at different modeling levels and implicitly shows the consideration of the abstraction levels which leads to a value of 1.5 from two of the whole dimension expressiveness which gives 75% against 25% for the second approach. These percentages serve as evaluation and comparison for determining the approach focus regarding the proposed dimensions. In similar fashion, the other dimensions are analyzed and compared accordingly. In summary, the first approach focuses on modeling levels of the architecture whereas the second is based on abstraction levels mainly used for reducing evolution complexity. It is noteworthy that the two approaches show identical values of object, type, stakeholder and time of evolution and both support the 'reduce' operating mechanism of evolution.

3.2 Framework assessment

Three prominent existing studies have been selected and surveyed to further evaluate the conceptual framework using the proposed metrics on their set of surveyed papers as follows:
The first considered survey study that involves 32 research papers was conducted by Ahmad [18] which focuses on approaches wherein changes impact the architectural level when analyzing and improving software evolvability. The investigation of the existing methods or techniques, either for systematic application or for empirical acquisition of architectural knowledge, categorizes evolution reuse knowledge into six broad themes: i) evolution styles, ii) change patterns, iii)

Table 2. Comparison of Two Evolution Approaches Using the Framework

Model		Oussalah (2006)		Barnes (2014)	
Object	Artifact	1	50%	1	50%
	Process	0		0	
Levels	Modeling	1	75%	0	25%
	Abstraction	0.5		0.5	
stakeholders	Architects/Designers	1	100%	1	100%
Time	Run	0	50%	0	50%
	Design	1		1	
OME	Reduce	1	50%	0.5	25%
	Emergence	0		0	
Type	Main-forms	1	12.50%	0.5	12.50%
		0		0.5	
		0		0	
		0		0	
	Main-categories	1	16.67%	0.5	16.67%
		0		0.5	
		0		0	

adaptation strategies and policies, iv) pattern discovery, v) architecture configuration analysis, and vi) evolution and maintenance prediction. The proposed thematic classification focuses on both time of evolution and type of evolution for reuse knowledge.

The second study by Breivold *et al.* [2] where five main categories of themes are identified based substantially on research topics through an investigation of 82 research papers: i) techniques supporting quality consideration during software architecture design ,ii) architectural quality evaluation, iii) economic valuation, iv) architectural knowledge management, and v) modeling techniques. A set of specific characteristics is provided with a view to refine each category to subcategories reflecting a common specification in terms of research focus, research concepts and context.

The third by Chaki *et al.*[12] recommends three classes of common type for architectural evolution:

- Maintenance focused evolution- these are works concerned with the use of correction decisions and architectural modifiability to address architects' concerns. These works often require intervention at the lowest modeling level.
- Open evolution- refers to all software architectural evolution works whose final architecture is not known a priori. These approaches infer, from an initial architecture, one or more solutions in respect of a context known beforehand.
- Closed evolution- Categorizes works in which the target architecture of the model is known before proceeding to the evolution of the initial architecture. It is a question of finding a sequence of operations that may guide an initial architecture to a desired one.

It is worth emphasizing that the first step was to devise a set of projections for the conceptual benchmarks and afterwards for the proposed taxonomy. These projections aim to reveal the coverage of each of the benchmarks that appear most frequently across the range of architectural evolution studies. Table 3 summarizes the quantification using weighted results of the six dimensions. The percentage of each dimension is calculated by dividing the sum of expressed studies by the total number of investigated studies. Detailed illustration for the provided results in Table 3 relating to the *Type* dimension (including *Forms* and *Categories*) are further shown in Table 4. The results presented in both tables reveal the just proportion (or disproportion) between weights granted for each proposed dimension.

Table 3. Coverage percentages of the proposed dimensions.

Dimensions	Studies	Ahmad 32	Breivold 82	Chaki 5	Weighted Average %
Object	Artifact	66.67	74.60	20	70.17
	Process	33.33	25.40	80	29.83
Levels	Modeling	86.67	88.23	64.29	86.80
	Abstraction	13.33	11.77	35.71	13.20
stakeholders	Architects/Designers	100	100	100	100
Time	Run	34.62	24.44	22.22	27.08
	Design	65.38	75.56	77.78	72.92
OME	Reduce	96.78	94.29	100	95.20
	Emergence	3.22	5.71	0	4.80
Type	Main-forms	10.59	41.86	100	35.89
	Main-categories	89.41	58.14	0	64.11

Weighted average $= \frac{\sum percentage_i \times n_i}{\sum n_i}$ *where* n_i *is the number of the surveyed papers.*

Table 4. Details of Form and Category Percentages.

Studies	Nbr	Main-forms %				Main-Categories %		
		Open Break	Open Seam	Close Break	Close Seam	Corr.	Perf.	Adap.
Ahmad	32	04.71	03.53	2.35	0	37.65	28.24	23.53
Breiold	82	36.05	04.65	1.16	0	24.42	26.74	6.98
Chaki	5	80.00	00.00	0.00	20	-	-	-

4 Discussion

Significantly, Table 3 shows that over three-quarters of the selected studies have explicitly fostered modeling levels for all works, which explains these studies' relevance to the architecture evolution topic. Further, percentages show that a wide

range of studies are committed to artifact as an object of evolution, except for [12] which focuses on process evolution studies. This is due to the fact that both existing evolution support formalism (SAEV [16], Query/View/Transformation-based [19], ...) and architecture description languages (ADL, UML, xADL, ...) for evolution models are all artifact oriented. It is worth noting that styles and patterns are actually more suitable for evolving architecture elements (artifacts) but contribute little to address the evolution issue of the styles themselves [20]. In addition, the modeling level has attracted major interest from the evolution community for the reason that it overlaps with the abstraction level concept. In the same way, an overwhelming percentage deal with reducing evolution from higher to lower level, i.e. top-down hierarchy modeling fashion. In our view, this preponderance is mostly due to a deductive reasoning influence which promotes the top-down method (i.e. reduce OME). However, this reductionism may deprive software architecture systems to introduce new solution opportunities according to the current and/or future environment assumptions. Regarding the temporal dimension, values are systematically in favor of the design-time which in our opinion reflects the interest to the anticipation activity to deal with evolution. It is noted that all these studies specify the importance of stakeholders requirements when dealing with evolution. The percentages in Table 4, show that the main categories sub-dimension attracted an explicit interest from all the studies apart from Chaki et al (2009).

5 Conclusion

In this paper, a conceptual framework is presented for modeling the classification of software architecture evolution approaches. This model is based on evaluating six proposed dimensions on a given approach. The proposal is part of a wider strategy to evaluate tendency of existing research within software architecture evolution according to well-defined dimensions. Thus, the proposed model provides architects with the capacity for understanding and analyzing the evolution before making technical choices. Considering the impact of evolution process through the hierarchical levels, an evolution framework of six different dimensions is devised. This would help to sketch the interior design of the architectural evolution solution space. Based on an experimental study of three well known classifications, the investigation of the coverage in software architecture evolution has drawn a number of conclusions on opportunities, strength and weakness of existing approaches. The obtained results highlight the lack of emergence techniques, which could be a very promising track. The proposed conceptual framework enjoys the merits of (i) compare different architectural evolution approaches, (ii) determine the appropriate evolution approach according to the dimensions that seems the most relevant to the application class and (iii) to guide thinking within research teams on new architectural evolution approaches.

References

1. Jazayeri, M.: Species evolve, individuals age. In: Principles of Software Evolution, Eighth International Workshop on, IEEE (2005) 3–9
2. Breivold, H.P., Crnkovic, I., Larsson, M.: A systematic review of software architecture evolution research. Information and Software Technology **54**(1) (2012) 16–40
3. Williams, B.J., Carver, J.C.: Characterizing software architecture changes: A systematic review. Information and Software Technology **52**(1) (2010) 31–51
4. Buckley, J., Mens, T., Zenger, M., Rashid, A., Kniesel, G.: Towards a taxonomy of software change. Journal of Software Maintenance and Evolution: Research and Practice **17**(5) (2005) 309–332
5. Zachman, J.A.: A framework for information systems architecture. IBM systems journal **26**(3) (1987) 276–292
6. Pigoski, T.M.: Practical software maintenance: best practices for managing your software investment. John Wiley & Sons, Inc. (1996)
7. Shaw, M., DeLine, R., Klein, D.V., Ross, T.L., Young, D.M., Zelesnik, G.: Abstractions for software architecture and tools to support them. Software Engineering, IEEE Transactions on **21**(4) (1995) 314–335
8. Bézivin, J.: La transformation de modèles. INRIA-ATLAS & Université de Nantes (2003) 13
9. Oreizy, P., Taylor, R.N.: On the role of software architectures in runtime system reconfiguration. IEE Proceedings-Software **145**(5) (1998) 137–145
10. Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R., Jazayeri, M.: Challenges in software evolution. In: Principles of Software Evolution, Eighth International Workshop on, IEEE (2005) 13–22
11. Kniesel, G., Costanza, P., Austermann, M.: Jmangler-a framework for load-time transformation of java class files. In: Source Code Analysis and Manipulation, 2001. Proceedings. First IEEE International Workshop on, IEEE (2001) 98–108
12. Chaki, S., Diaz-Pace, A., Garlan, D., Gurfinkel, A., Ozkaya, I.: Towards engineered architecture evolution. In: Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering, IEEE Computer Society (2009) 1–6
13. Oussalah, M., et al.: Génie objet: analyse et conception de l'évolution. Hermès Science publications (1999)
14. Terho, H., Suonsyrjä, S., Systä, K., Mikkonen, T.: Understanding the relations between iterative cycles in software engineering. In: Proceedings of the 50th Hawaii International Conference on System Sciences. (2017)
15. Brooks, R.A.: A robot that walks; emergent behaviors from a carefully evolved network. Neural computation **1**(2) (1989) 253–262
16. Oussalah, M., Sadou, N., Tamzalit, D.: Saev: A model to face evolution problem in software architecture. In: Proceedings of the International ERCIM Workshop on Software Evolution. (2006) 137–146
17. Barnes, J.M., Garlan, D., Schmerl, B.: Evolution styles: foundations and models for software architecture evolution. Software & Systems Modeling **13**(2) (2014) 649–678
18. Ahmad, A., Jamshidi, P., Pahl, C.: Classification and comparison of architecture evolution reuse knowledgea systematic review. Journal of Software: Evolution and Process **26**(7) (2014) 654–691
19. Mens, T., Van Gorp, P.: A taxonomy of model transformation. Electronic Notes in Theoretical Computer Science **152** (2006) 125–142

20. Hassan, A., Oussalah, M.: Meta-evolution style for software architecture evolution. In: International Conference on Current Trends in Theory and Practice of Informatics, Springer (2016) 478–489