Evolution Styles: Multi-View/Multi-Level Model for Software Architecture Evolution

Adel Hassan*, Mourad Oussalah
LS2N, University of Nantes, 44322 Nantes, France.

* Corresponding author. Tel.: +33251125834; email: Adel.hassan@eut.univ-nantes.fr
Manuscript submitted December 04, 2017; accepted March 22, 2018.
doi: 10.17706/jsw.13.3.146-154

Abstract: With the ever growing complexity and size of software systems, the role and responsibilities of architect have changed extensively. Architecting has become a broad, creative activity that involves discovering stakeholder concerns, evaluating alternatives, making tradeoffs and planning the software process, rather than developing a simple artifact just for capturing information. Therefore, the architect is in greater need of models, methods, techniques and tools to assist in the planning and conduct of software architecture evolution. To this end, we propose evolution styles as a multi-view methodology for software architecture evolution modeling. The styles are focused on how to develop an evolution model that can capture the best architectural evolution knowledge and practices in a particular domain, and how to demonstrate this knowledge in multiple forms in order to cover the different stakeholders' viewpoints. To show the applicability of our approach, we have developed a prototype for a multi-view and multi-abstraction evolution styles editor.

Key words: Software evolution, software architecture evolution, evolution styles, multi-view modeling.

1. Introduction

Today's information systems are becoming more and more complex, with most of them being software-intensive systems. They often include large-scale heterogeneous distributed software components, embedded systems, telecommunications, wireless ad hoc systems and COTS. As the markets and technology are continuously changing, these systems need also to change in order to embrace the new business opportunities, customer requirements and technological infrastructure. Whenever a system needs to be changed, the ideal starting point of the evolution team is to understand the system and then attempt to find a suitable set of modifications. In this context, planning the large-scale evolution is a highly challenging activity that prerequires an understanding of the overall system structure, consideration of design rationale, and principled trade-offs among various qualities of concern [1].

It was measured that, a system comprehension consumes more than half of the time and effort of the evolution process [2]. Software architecture represents the high-level structure of the software system, views the design decisions and abstracts far away from the complexity of low level details [3]. With an increase in the size and the complexity of the software system, the academic and industrial computing community acknowledges the importance of software architecture as a central artifact, that can guide and plan the process of the software system's construction and evolution.

In the context of evolution process, the software architecture has been considered from two points of

view. On the first hand, software architecture can be handled as an artifact for evolution, the aim of which is to help the architect plan and restructure the system. On the other hand, software architecture can be handled as an artifact of evolution, because it must be evolved in order to preserve the consistency between the architecture and the source code [4].

Moreover, as the software architecture captures early design decisions that had a significant impact on the quality of the system under evolution, it is important, if not essential, to understand those decisions as early as possible in the evolution process. However, architecture evolution is about making new design decisions, or removing obsolete ones, in order to satisfy new requirements. It concerns the changes applied on architecture, including their components, connectors, or their configuration.

Evolution is a complex process that cannot be completed over night. Divers groups of stakeholders are involved in this process, each of which performs a certain task. Thus, architects are in need of models, methods, techniques and tools that allow them to guide and plan this process as well as enable them to efficiently communicate with the different categories of stakeholder.

Thus, architecture evolution has become an active area of research on which a number of software architecture researchers have focused their attentions. These attentions have been presented in some of the interesting semantic literature reviews and in comparing studies [5] and [6]. One field of architecture evolution research concerned methods, tools and techniques that can assist architects to plan and conduct the software architecture evolution. An evolution styles approach belongs to this category, its aim being to define an evolution model that can embraces the best practices and knowledge in a particular domain of architecture evolution [7]. It was firstly introduced by Oussalah *et al* [8], after which a number of evolution styles were proposed [1], [4], [9], [10], each of which defines its own methodology in accordance with a certain viewpoint to model the architecture evolution process. This work presents an attempt to embrace the multi-view methodology in software architecture evolution modeling, in an endeavour to reduce the complexity and to provide several views, each of which covers a relevant set of aspects or satisfies certain stakeholders' concerns.

The remainder of this paper is structured as follows. Section 2 provides a background and the related work of software architecture evolution and evolution styles. Section 3 discusses the multi-view methodology and presents its contributions in the domain of software architecture. Section 4 present a conception of the construction of a multi-view evolution style and how this can be expressed in different abstraction levels. Section 5 proposes an extension of MES to embrace the concepts of multiple views. Section 6 gives a brief overview on our ongoing work, a prototype tool for evolution styles editor based on the multi-view methodology. Finally, the contributions of the paper are summarised Section 7.

2. Background and Related Work

Knowledge is essential in all types of work. Everyone knows how they should carry out their work, but sometimes this knowledge needs to be reused and shared with others. A process model aims to capture this knowledge and abstract in order to contain the necessary information that is required to perform a certain task or satisfy certain needs. In general, several domain-specific modeling languages DSMLs [11] have been devised for different purposes that have brought many benefits such as managing the complexity, preserving and reusing of expert knowledge. The diversity of modeling languages is due to the different perspectives of modeling, which often require different concepts with different properties [12], as well the particularity of some domains.

Evolution styles aim to model software architecture evolution activities in order to equip the architect with a body of knowledge that encapsulates best practices and knowledge in a particular domain of architecture evolution. This body (library) of domain-specific architecture evolution knowledge can be

enhanced with facilities such as save, search, retrieve and compare information, in order to allow the architect to better retrieve and use the suitable knowledge from this library (which best fits the current context under evolution).

Garlan et al. [1] have presented a proposal of evolution styles that express the software architecture evolution as a set of potential evolution paths from the initial architecture and the target architecture. Each path can comprise a sequence of evolution transitions, each of which is specified by means of evolution operators that determine how the architecture can be modified. Evaluation functions have been defined to assess and compare evolution paths with respect to certain properties.

Cuesta et al. [4] have proposed Architectural-Knowledge-driven evolution styles (AKdES), which are based on architecture design decisions whenever an evolution step (on an evolution path) is made. This is a result of evolution decisions. Saving all these architecture decisions in an architectural knowledge allows us to use this knowledge as a chronological view that composes information (alternatives, architectural design decisions, rationale) about the possible set of, or sequence of, evolution steps which can be carried out to reach the intended design.

Oussalah et al. [9] define evolution styles, the aim of which is to capture and transfer architecture knowledge in domain-specific evolution. They describe the software architecture evolution as generic transformations (evolution patterns). These generic transformations describe architecture changes as a series of steps (evolution operations), which can be instantiated into a lower abstraction level (with more details) in order to customize or fit a certain context of evolution.

In our previous work [7] based on the meta-modeling technique, we propose an evolution meta-style MES, the aim of which is to define the necessary concepts in order to express evolution styles. With these core concepts, several perspectives can be expressed using the same meta-concepts. The conceptual model (MES) was validated and conformed by different styles. Paper [13] represents an extension of MES in order to annotate the dynamic evolution concepts. It handles some issues relevant to the dynamic evolution such as safe-stopping, state transfer and dynamic scheduling.

Here, as MES is a component-oriented framework, we aim to stimulate architectural viewpoints when modeling the evolution process. As the architecture model often expresses several viewpoints; structural, behavioral and decisions knowledge, we aim to investigate how this multi-view methodology can be embraced by MES in order to develop a multi-view evolution style.

3. Multi-view Modeling

A model is an abstract representation of reality (phenomenon). Often, there are several categories of interest groups in any phenomenon. Each group is concerned about certain aspects of the phenomenon, and look for a model that can cover their concerns. It is almost impossible to capture all aspects of the design, which cover different stakeholders' viewpoints, in a single description. Most design methodologies endeavor to develop an interlocking set of views, each of which intends to cover one or more relevant aspects of the design [14].

The multi-view modeling methodology has been introduced in several software engineering areas, such as enterprise and software architecture, requirement engineering, software process and system modeling, and development methodologies [15]. Each discipline specifies its own viewpoints; they share the basic idea of decomposing a complex artifact model into multiple views.

Since our approach proposes an architecture-centric methodology for modeling an architecture evolution process, we are more interested in reviewing the multiple viewpoints in architecture model.

Perry and Wolf [16] during the early days of software architecture, proposed one of the most interesting definitions of software architecture as a 3-tuple consisting of Elements, Form, and Rationale.

These three parts formulate the essence of the architecture model, as the extending forms of Taylor, Medvidovic and Dashofy [3], whereby the meaning of What, How and Why questions are embraced in this form: What are the elements of the system? (Structural view). How are the elements organized and how do they interact? (Behavioral view). Why are particular elements used? Why are they combined in a particular way? (Decision view).

Therefore, the architecture's models, method and tools have been proposed to cover one or all of these viewpoints. For example, in the first generation of Architecture Description Languages (ADLs), the system structure (structural view) was the basic aspect that was captured by most of these languages as a component-connector view, such as, Aesop, Wriigh and UniCon [17].

The dynamic ADLs (such as Darwin and Rapide, dynamic Wright [18]) come to cover the system behavior in order to capture the interaction among system elements, or between the system and its environment. A formal behavioral modeling method (e.g. algebra, Petri nets and Z nations) have been used to cover the proprieties of this view.

Finally, with the growing complexity and size of software systems, the computing community and the organizations acknowledge the importance of software architecture. Several approaches have been published focusing on the Rationale element (Decision view), which aims to cover the reason behind design decisions. Jansen and Bosch emphasized this view, which manifested in their definition of software architecture as a composition of a set of architectural design decisions [19]. Several frameworks to document the architectural design decision have been proposed, and Capilla et al. [20] have summarized the academic and industrial achievement in this trend.

Therefore, an architectural model of software architecture evolution process can be viewed in accordance with these viewpoints.

4. Multi-view and Multi-abstraction Evolution Styles

As mentioned in the previous section 3, multiple views aim to decompose the complexity that exists by stuffing all the design details in one model. Moreover, some views (certain aspect) attract the interest of different stakeholders' categories, each of which needs a certain level of detail about this view.

For example, UML provides 13 diagrams for a system and process modeling classified in two main categories: structural and behavioral. There are three structural diagrams that are architectural-oriented diagrams which are: component, deployment and package diagrams, each of which provides a different level of detail about the structural view of the system architecture. The component diagram can provide greater detail about a certain package. Moreover, Heesch *at al.* [21] proposed a framework to view the architecture design decision, which comprises four forms, which are: a Decision Detail viewpoint, a Decision Relationship viewpoint, a Decision Chronology viewpoint, and a Decision Stakeholder in order to satisfy different concerns of the stakeholders about this view.

In this respect, evolution styles can be expressed in multiple views, whereby each view can be expressed in different abstraction levels, each which provides a certain level of detail. Figure 1 shows a matrix to illustrate the concepts of multi-view/multi-abstraction evolution styles.

Evolution style: refers to a modeling approach, the aim of which is to capture the main characteristics of a set of activities performed in a domain-specific software architecture evolution, in order to define the vocabulary of concepts necessary to model this process in accordance with a certain viewpoint (aspects, concerns) [7].

View: refers to a facet of an evolution style from which a set of relevant aspects or stakeholder concerns can be separately expressed. An evolution style might be composed of several views. For example, as MES is an architectural-oriented modeling methodology, an evolution style model can be expressed from different

architectural views, which are: structural, behavioral or decision knowledge view whereby different levels of detail about a certain view can be reached at different abstraction levels.

Abstraction levels: refer to a set of detail levels that the view provides. A view might be presented at several abstraction levels. These multiple levels aim to cover different amounts of detail that need to satisfy the curiosity of different stakeholders' categories, or those that have a certain interest in the view.

Interaction: refers to the link that can express the relations between different abstractions which can be classified in two categories:

- ***Inter-view link***: defines the relation between two distinct abstractions, which belong to two different views.
- ***Intra-view link***: defines the relation between two distinct abstraction levels, which belong to the same view.

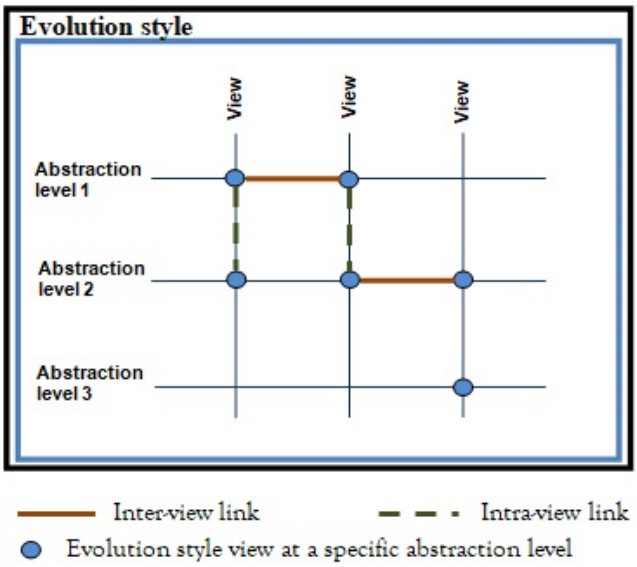

Fig. 1. A conceptual matrix for multi-view & abstraction evolution style.

5. Multi-view/abstraction Evolution Meta-Style

MES in [7] was defined as Meta-meta-model for the domain of architecture evolution. It is a component-based framework for architecture evolution process modeling. The approach is compatible with the four modeling levels of the OMG, whereby each evolution model is an instance of the model in the level above (its Meta), including the meta-styles MES, which is an instance of itself.

MES can be refined to embrace the concepts of multi-view and multi-abstraction methodology, thus the aim of this work is not to define a new meta-style, but to annotate MES with concepts of this methodology that are represented in the previous section.

Thus, Fig. 2 illustrates our proposition to extend MES with the necessary concepts to satisfy the requirement of defining the multi-view modeling methodology in the domain of architecture evolution (white boxes refers to MES elements; grey boxes refer to proposed additional elements).

However, any evolution style can be expressed by using instances of the main concepts of MES. The style should have at least one view that covers one or more relevant aspects of the architecture evolution process in accordance with a certain stockholder viewpoint.

The view could provide different levels of details. Therefore, an evolution styles tool should come with one or more diagrams, each of which can be used to edit a certain view at a certain level. A specific dialogue

or a template (not a distinct diagram) can be also used to provide more information about a certain notation in a diagram.

Role: It represents a set of responsibilities, e.g. it describes the required skills, tools and techniques that are required to perform a specific architecture evolution operation.

Operation: One of the units of process that produces visible changes in the architecture. The operation is associated with roles and architecture elements through the interaction elements.

Architecture element: The inputs and the outputs of an evolution operation. This may be an architectural style, package, component, connector, attachment, port, or interface, etc.

Interaction: An entity that governs the relationships between these elements and their behaviors. Generally, it determines what roles participate in an operation, what kind of inputs and outputs there should be, and how views and abstractions are interrelated.

Interface: A place in which elements interact and which determines rules that should be followed by elements in order to intercommunicate compatibly.

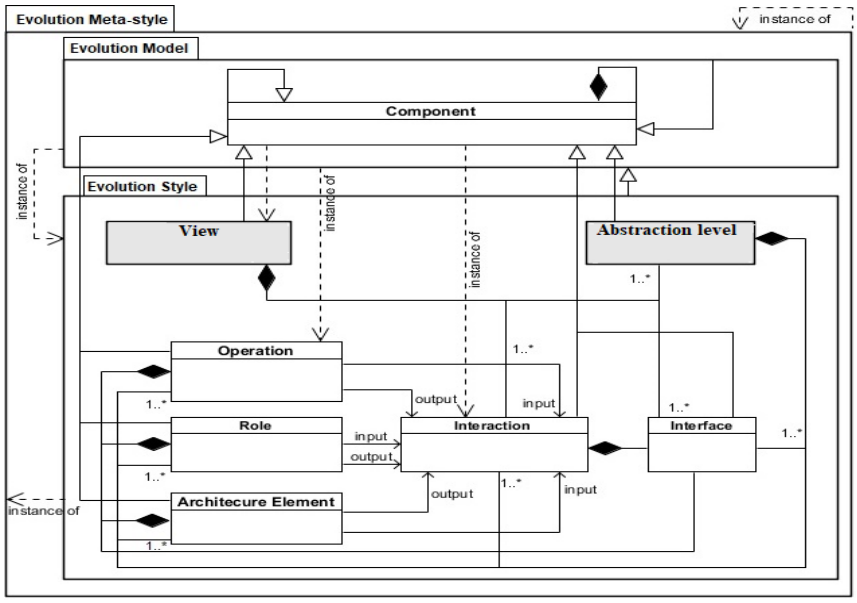

Fig. 2. MES +.

6. Prototype Tool

To validate our approach, we present a prototype of a multi-view and multi-abstraction evolution styles editor. The tool is based on the ADOxx meta-modeling platform [22].

ADOxx is a meta-modeling platform, the aim of which is to facilitate the design and implementation of modeling tools for domain specific languages (DSL). It is developed by the BOC Group, which was founded by Prof. Dr. Dimitris Karagiannis as a spin-off of the University of Vienna.

Several tools have been developed by the ADOxx platform for several domains. Moreover, some open source libraries (e.g. UML, BPMN) and code-repositories have been developed which can help and guide developers in designing their own tool.

Structure is a basic aspect that can be captured by most architecture models. Since, MES is an architectural-centric framework that aims to architect the architecture evolution, thus a structural view of evolution styles is the main artifact of the editor.

To this end, workflow is used as the paths' diagram to depict the possible ways by which to evolve architecture. A more abstract level of the structural view (paths diagram) can be used to provide more information about a certain notation in this diagram.

A UML component diagram and deployment diagram have been used to annotate more structural details about each node on the evolution paths.

By double clicking on a node, a description dialogue will be opened, whereby a component or deployment diagram can be associated with this node. The diagram name will appear as a link on the node in order to open this diagram by double clicking on the link.

This relation between the evolution path diagram and component, or deployment diagram represents the intra-view link type that has been introduced in section 4.

Fig. 3 shows a screenshot of the evolution styles editor based on the MES+ methodology. As we can see in the figure, from the mode we can select a certain evolution style approach that has the same view. For example, the structural view (diagram) can be depicted by different notations from different evolution style approaches. By selecting one style from the mode the suitable notations of this view will be displayed on the tool palette.

Future research will concentrate on enhancing the presented tool. Different evolution styles with different views and their abstraction levels will be included in the implementation. Supporting facilities, like a paths evaluation, will be provided using graphic analysis methods, and an evolution decision template will provide architectural evolution knowledge about each transition on the paths.

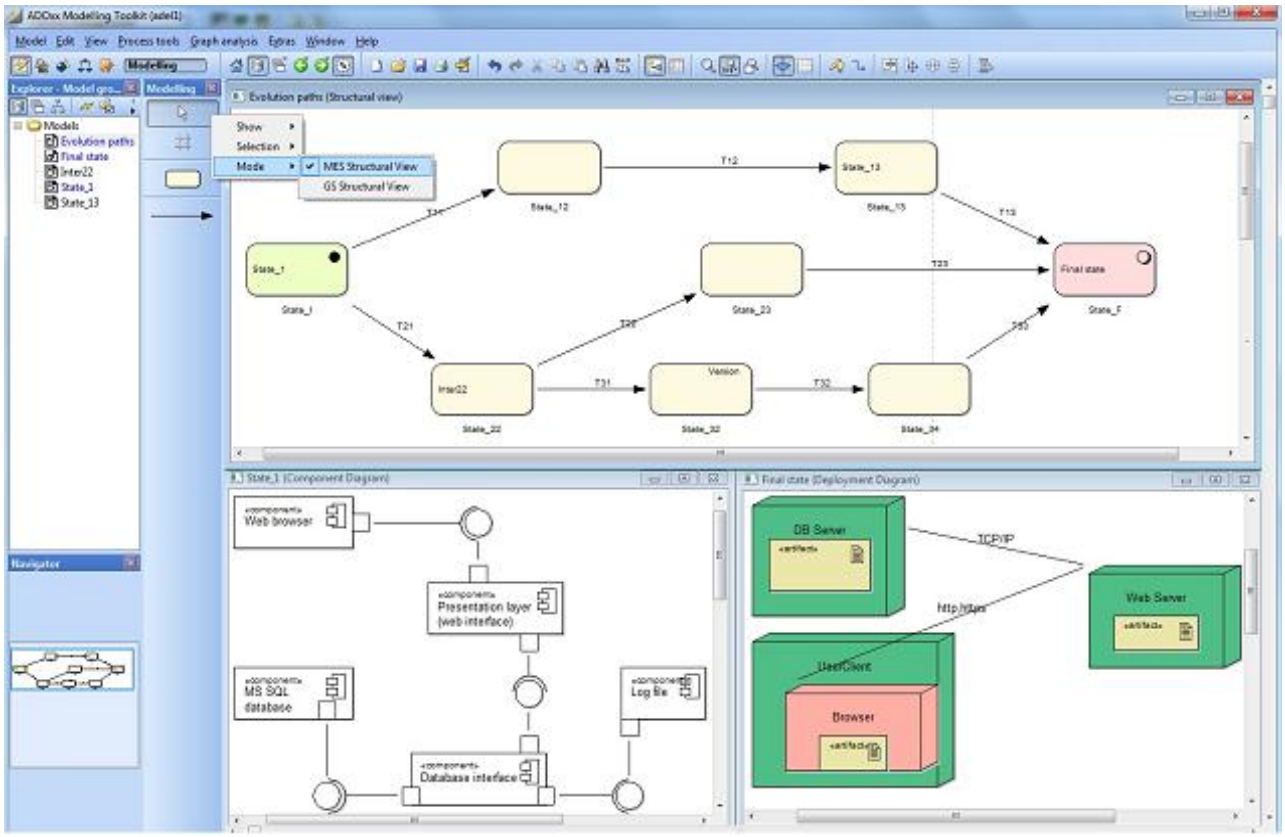

Fig. 3. A screenshot of evolution style editor based on MES +.

7. Conclusion

Software architecture expresses the design decisions of the system at a high level of abstraction, far away from the complexity of the low-level details. It is the backbone of the software systems and provides guidance on their construction and evolution. System reconstruction comprises complex activities that

require a deep understanding of the architecture design decisions in order to plan and implement the required change. Evolution teams, especially architects, are in needs of models, methods and tools that can support them in planning and conducting this process. This work proposes a multi-view modeling methodology for software architecture evolution, the aim of which is to capture and reuse the architectural evolution knowledge and practices in a particular domain. The results of this work may serve as a road map for architects that will help them to develop evolution styles with multiple views that can satisfy their needs.

References

[1] Barnes, J. M., Pandey, A., & Garlan, D. (2013). Automated planning for software architecture evolution. *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering* (pp. 213-223).

[2] Zelkowitz, M. V., Shaw, A. C., & Gannon, J. D. (1979). *Principles of software engineering and design* (1st ed). Englewood Cliffs: Prentice-Hall.

[3] Medvidovic, N., & Taylor, R. N. (2010). Software architecture: Foundations, theory, and practice. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2* (pp. 471-472).

[4] Cuesta, C. E., Navarro, E., Perry, D. E., & Roda, C. (2013). Evolution styles: Using architectural knowledge as an evolution driver. *Journal of Software: Evolution and Process*, *25*(*9*), 957-980.

[5] Breivold, H. P., Crnkovic, I., & Larsson, M. (2012). A systematic review of software architecture evolution research. *Journal of Information and Software Technology*, *54*(*1*), 16-40.

[6] Jamshidi, P., Ghafari, M., Ahmad, A., & Pahl, C. (2013). A framework for classifying and comparing architecture-centric software evolution research. *Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering.*

[7] Hassan, A., & Oussalah, M. (2016). Meta-evolution style for software architecture evolution. *Proceedings of the International Conference on Current Trends in Theory and Practice of Informatics* (pp. 478-489).

[8] Oussalah, M., Tamzalit, D., Le Goaer, O., & Seriai, A. D. (2006). Updating styles challenge updating needs within component-based software architectures.

[9] Le Goaer, O., Tamzalit, D., Oussalah, M. C., & Seriai, A.D. (2008). Evolution styles to the rescue of architectural evolution knowledge. *Proceedings of the 3rd international workshop on Sharing and Reusing Architectural Knowledge*.

[10] Oussalah, M., Sadou, N., & Tamzalit, D. (2006). SAEV: A model to face evolution problem in software architecture. *Proceedings of the International ERCIM Workshop on Software Evolution* (pp. 137-146).

[11] García-Borgoñon, L., Barcelona, M. A., García-García, J. A., Alba, M., & Escalona, M. J. (2014). Software process modeling languages: A systematic literature review. *Journal of Information and Software Technology*, *56*(*2*), pp.103-116.

[12] Kent, S., (2002). Model driven engineering. *Integrated formal methods* (pp. 286-298). Springer Berlin/Heidelberg.

[13] Hassan, A., Queudet, A., & Oussalah, M. (2016). Evolution style: Framework for dynamic evolution of real-time software architecture. *Proceedings of the Software Architecture: 10th European Conference, ECSA 2016, Proceedings 10* (pp. 166-174). Springer International Publishing.

[14] Boiten, E., Bowman, H., Derrick, J., Linington, P., & Steen, M. (2000). Viewpoint consistency in ODP. *Computer Networks*, *34*(*3*), 503-537.

[15] Kheir, A., Naja, H., Oussalah, M. C., & Tout, K. (2013). From viewpoints and abstraction levels in

software engineering towards multi-viewpoints/multi-hierarchy in software architecture. *Proceedings of the Eighth International Conference on Software Engineering Advances* (p. 478).

[16] Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, *17*(*4*), 40-52.

[17] Medvidovic, N., & Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on software engineering*, *26*(*1*), pp.70-93.

[18] Kacem, M. H., Jmaiel, M., Kacem, A. H., & Drira, K. (2005). Evaluation and comparison of ADL based approaches for the description of dynamic of software architectures.

[19] Jansen, A., & Bosch, J. (2005). Software architecture as a set of architectural design decisions. *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*.

[20] Capilla, R., Jansen, A., Tang, A., Avgeriou, P., & Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, *116*, 191-205.

[21] Van, H. U., Avgeriou, P., & Hilliard, R. (2012). A documentation framework for architecture decisions. *Journal of Systems and Software*, *85*(*4*), 795-820.

[22] Fill, H. G., & Karagiannis, D. (2015). On the conceptualisation of modelling methods using the ADOxx meta modelling platform. *Enterprise Modelling and Information Systems Architectures*, *8*(*1*), 4-25.

Mourad Chabane Oussalah is a full professor of computer science at the University of Nantes and the chief of the software architecture modeling Team. His research concerns software architecture, object architecture and their evolution. He worked on several European project (Esprit, Ist, ...). He is (and was) the leader of national project (France Telecom, Bouygues telecom , Aker-Yard-STX, ...)

He earned a BS degree in mathematics in 1983, and Habilitation thesis from the University of Montpellier in 1992.

Adel Hassan received his B.S. degree in computer science from University of Sebha, Libya. He obtained a M.S. degree in software engineering from University of Bradford, England. He is currently enrolled in the Ph.D. study in software engineering at university of Nantes, France. His current research interests are software architecture evolution modeling and evolution styles.