Architecture Description leveraging Model Driven Engineering and Semantic Wikis

Alessandro Baroni, Henry Muccini
Department of Information Engineering,
Computer Science and Mathematics
University of L'Aquila, Italy
{alessandro.baroni, henry.muccini}@univaq.it

Ivano Malavolta
Gran Sasso Science Institute
L'Aquila, Italy
ivano.malavolta@univaq.it

Eoin Woods
Artechra
Hemel Hempstead, Hertfordshire, UK
eoin.woods@artechra.com

***Abstract*—A previous study, run by some of the authors in collaboration with practitioners, has emphasized the need to improve architectural languages in order to (i) make them simple and intuitive enough to communicate effectively with project stakeholders, and (ii) enable formality and rigour to allow analysis and other automated tasks. Although a multitude of languages have been created by researchers and practitioners, they rarely address both of these needs.**

In order to reconcile these divergent needs, this paper presents an approach that (i) combines the rigorous foundations of model-driven engineering with the usability of semantic wikis, and (ii) enables continuous syncronization between them; this allows software architects to simultaneously use wiki pages for communication and models for model-based analysis and manipulation. In this paper we explain how we applied the approach to an industry-inspired case study using the Semantic MediaWiki wiki engine and a model-driven architecture description implemented within the Eclipse Modeling Framework. We also discuss how our approach can be generalized to other wiki-based and model-driven technologies.

I. Introduction

Software architects create architecture descriptions (ADs) throughout a system's life time for a range of purposes, including documenting and communicating design decisions to the various stakeholders, as a basis for analyzing and evaluating alternative architectures, to support system planning, scheduling and budgeting activities, and to meet many other needs (see [12, ∮4.4] for a comprehensive list).

A primary finding from an industrial study conducted with practitioners to analyze the perceived strengths, limitations and requirements of architectural languages [14] emphasizes the practitioners' requirement that languages can be both *simple and intuitive* to allow effective communication among stakeholders, and yet also *formal and structured* to allow the possibility of analysis and other automated tasks. These two distinct requirements reflect the so called "extrovert" (i.e., communication-oriented) and "introvert" (i.e., analytical) nature of the work of software architects. Yet, despite this clear need, most academic research efforts so far have mainly focused on formal and domain-specific architectural languages (which compromise on the communication requirements) or on extending the UML (compromising the analytical requirements), while industry turns to informal languages that are easy to use and communicate far more effectively, but cannot support the analytical needs of architects. Therefore, a major divergence between research and real needs is evident.

Due to their accessibility, effectiveness for information sharing, and ease of use, wiki-based architecture descriptions are quite popular in industrial contexts [7]. However, architecture models are also being used for a multitude of other purposes, such as assessment and functional and non-functional analysis (e.g., performance and reliability, testing, conformance checking [2]).

In this paper, we propose an approach that allows software architects to (i) track and share informal (and loosely structured) architectural knowledge by means of semantic wikis, (ii) maintain and analyse the architectural design by means of well-defined MDE-based architecture models within their own tools, and (iii) automatically and continuously synchronise the information between the wiki and MDE-based models. The approach builds on the technical foundations of model-driven engineering (as the analytical model) and on the communication-oriented nature of wiki-based documents.

The remainder of this paper is organized as follows. Section II of the paper provides relevant preliminary information. Section III presents our proposal from a conceptual perspective, followed by a description of the technology-specific implementation. Related work is discussed in Section IV, while Section V discusses advantages and current limitations of our approach, and its generalization to other technologies. Finally, conclusions and future work are presented in Section VI.

II. Background

A. MDE for Architecture Description

Model-Driven Engineering (MDE [13]) is a promising approach for effectively expressing domain concepts by creating abstractions of selected aspects of a system and considering specific properties of the system early in the life cycle. In MDE, domain-specific modeling languages (DSMLs) are used to describe the application; they are defined using metamodels, which define both the relations between concepts within the domain and their semantics. DSMLs are used to build a model of the system according to the semantics and constraints defined in their metamodel (in this case the model is said to "conform to" the DSML metamodel). A given system may have *n* different models, each model representing a specific aspect of the system (e.g., requirements, design specification, or even the program code).

In software architecture, MDE can be used to achieve many different goals. For example, metamodels are used to define the concepts of architectural languages [10], model transformations are used to transform architecture models among

ALs [6], to define architectural viewpoints for architecture reconstruction [9] and to build architecture frameworks according to the IEEE/ISO/IEC 42010 standard [11]. In this work, an architectural language can be considered as a metamodel with its own tool, methodology and process, and an architectural model is a model conforming to the architectural language metamodel. By applying the MDE principle that *models are precise artifacts that can be understood by computers and can be automatically manipulated*, this work aims to enable the automatic exchange and synchronization of architectural information across MDE-based tools and semantic wikis.

B. Semantic Wikis and Ontologies

A wiki is a web-based system whose content is collaboratively added, updated, and organized by its users. With the addition of an ontology, a semantic wiki "extends the application area of a normal wiki by providing improved navigation and search, context dependent presentation, etc."[16]. A high-profile example that shows the result of applying these features is Wikipedia, where a the growing number of pages offer structured content (and need constant maintenance) such as 'List of urban areas by population" or 'List of banks in Europe". This metadata availability provides a good framework to allow the exchange of data with external applications that share the same ontology.

In the field of software architecture, wikis have primaryil been used to produce architecture descriptions [4] and to manage architectural knowledge throughout the system life cycle [5]. This work exploits semantic wikis as tools to record architectural knowledge, and leverages the semantic technologies of wikis to allow automatic extraction of relevant information from them.

III. The Proposed Solution

A. Concepts

Keeping track of the architecture knowledge base for a system (e.g., requirements, ideas, context information, design decisions, etc.) is important for communication purposes but this is only part of the story. Beyond this, architecture models are also a special kind of architecture knowledge element which can be used to capture the design decisions that have been made and evaluate them by means of specific tools (e.g., doing performance analysis of the architecture or checking whether the architecture conforms to a reference architecture).

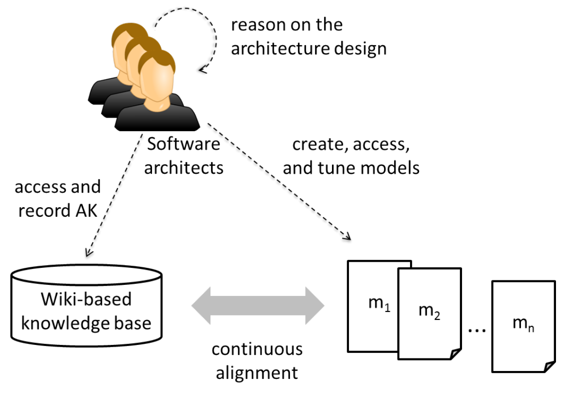
reason on the architecture design
Software architects
access and record AK
create, access, and tune models
Wiki-based knowledge base
continuous alignment
m1
m2
...
mn

Fig. 1. Usage scenario of our solution

As explained later in Section IV, a system's architecture knowledge base and its architecture design models are usually created separately using different tools and are seldom linked. This implies extra effort and risk for software architects, who have to manually synchronise the information in the knowledge base with the models created using tools. Figure 1 shows the solution we propose to this problem. Fundamentally, we propose to use semantic wiki pages as communication-oriented artifacts, for recording and sharing architecture knowledge, the use of MDE-based architecture models (M_1, M_2 and M_n in the figure) to support analysis and design, and the automatic synchronisation of information between the two. This kind of separation allows software architects to:

- access and record the software architecture knowledge base in a semi-structured manner (i.e. using the structure dictated by an ontology underlying the wiki);
- create, access, and tune the MDE-based architecture models directly in their corresponding tools (e.g., AADL models can be created, analysed, and manipulated in the OSATE2 tool set[1]);
- reason about the current architecture design both by using the architecture models in their corresponding tools and by referring to the wiki-based architecture knowledge base *at the same time*.

For what concerns the usage process of our approach, we identify two main roles: (i) the *MDE expert* is in charge of acquiring the metamodel representing the architectural language and in setting up the linked semantic wiki, and (ii) *software architects* are in charge of reasoning on the system being developed, accessing, and recording architectural knowledge in either the semantic wiki or MDE-based architecture models. It is important to note that the architectural knowledge in both the semantic wiki and the various MDE-based models is automatically and continuously synchronised using the synchonisation engine developed as part of this work. The engine achieves this by creating and maintaining a set of relationships between the metamodelling technical space and the ontological technical space.

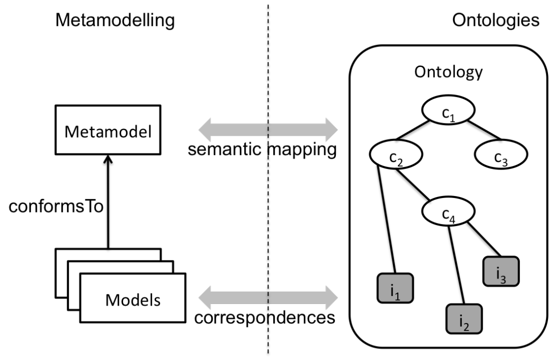

Fig. 2. Conceptual overview of our solution

As shown in Figure 2, the metamodelling technical space is concerned with the MDE-based architecture models, their metamodels and all the other artifacts within the metamodelling stack (see Section II-A). The ontological stack concerns ontologies, their classes, individuals (i.e., instances of ontological classes), and so on. The relationships between the metamodeling and the ontological technical spaces exist at two different levels of abstraction, specifically:

- the *semantic mapping*, which relates classes, attributes and references in a metamodel with classes, properties and

[1]https://wiki.sei.cmu.edu/aadl/index.php/Osate_2

associations in its corresponding ontology (c_1, c_2, c_3, and c_4 is shown in Figure 2). Without explaining all of the details, in this part of the solution we build upon existing work on integrating metamodels and ontologies [15] which allows us to use a model transformation t which is able to produce a set of ontological classes starting from a metamodel. A summary of the mappings implemented by t is summarized in Table 3. It is worth noting that many semantic wiki engines are able to generate wiki page templates reflecting the structure of the classes belonging to their underlying ontology. This allows software architects to exploit that structure to semantically organize architectural information in accordance with the concepts of the architectural language being used for producing the MDE-based models of the architecture being developed. Also, the presence of bidirectional transformations does not limit the kinds of models being represented.

Ontology	**Metamodel**
ontology	package
class	class
association	reference
attribute	attribute
data types	data types
subclass	subclass
enumeration	enumeration
multiplicity	cardinality

Ontology	**Model**
individual	object
attribute values	attribute values

Fig. 3. Mapping ontologies and (meta) models

• the *instance-level correspondences* consists of a set correspondences between (i) the objects within the models (conforming to the above mentioned metamodel) and (ii) individuals (i_1, i_2, i_3 in Figure 2) within the corresponding ontology . A high-level overview of those correspondences is summarized in Figure 3. As part of this work, we developed a module which is able to automatically enforce these correspondences across ontological entities in the semantic wiki and their corresponding objects within the MDE-based models. Clearly, given the different purposes of semantic wikis and MDE models, we expect that only a subset of the information contained in the wiki will need to be synchronized with MDE models, and only a subset of the information in the models will need to be synchronised with the wiki.

Once an ontology has been generated from the metamodel of the AL, our engine is able to synchronise the architectural information across the two technical spaces. By building on the semantic wiki, our engine extracts the relevant information from the semantically structured data in order to continuously synchronize the wiki pages and the MDE-based models. Moreover, it is important to note that instance-level correspondences are *bidirectional* and *continuous*; that is, our approach is able to instantly reflect updates performed on a wiki page back to the MDE-based architecture models, and vice versa.

B. Implementation

The first prototype of our approach[2] is implemented by building on the Eclipse platform. We decided to use Eclipse because many extensions already exist covering some aspects of our approach; in particular, metamodels can be defined using the Eclipse modelling Framework (EMF), a Java framework and code generation facility for MDE-based tools. Also, many architectural languages (in particular AADL and UML) are supported by the EMF modelling framework. For the semantic wiki engine, we chose to use MediaWiki[3] and its Semantic MediaWiki plugin (SMW[4]). This decision was based on the fact that SMW is very well documented, widely used, and has a very active support community both in academia and in industry. However the approach is not dependent on MediaWiki and a plan to generalize it is outlined in Section V.

From an abstract point of view, our prototype implementation is made up of two main components. The **Wiki2Model** component is responsible for updating the architecture models every time it receives a new update operation from the Wiki. Update operations are created for each change to portions of wiki pages that represent at least one element in one of the MDE model (i.e., pages belonging to model-related categories in the wiki). The **Model2Wiki** component is responsible for updating the wiki pages when updates are made to the corresponding architecture models. When a new version of an architecture model is produced this component compares the new version of the modified model with the previously synchronized one and generates a set of update operations to be propagated to the MediaWiki engine corresponding to the changes in the model.

IV. Related Work

ADDSS, the Architecture Design Decision Support System [3], was one of the first solutions to exploit web-based technologies for recording architecture-related information. ADDSS is based on a metamodel and a web-based tool that allows software architects to record, maintain and manage design decisions, requirements and other information such as architectural styles and patterns. Architecture models can be added to the ADDSS tool as plain figures.

PAKME [1] is a web-based knowledge management tool for capturing design decisions and their contextual information and recording them according to a data model tailored to the software architecture domain. The main focus of PAKME is on requirements (via scenarios), architectural design decisions and architectural patterns; it also contains a wiki-based component to support collaborative decision making.

The authors of [8] explored the applicability of wikis for supporting software architects in their typical architecting activities. Wikis performed well in managing non-architectural knowledge, in supporting the integration with other tools and in providing intuitive interfaces with a shallow learning curve. However, an important limitations found was the difficulty of codifying architectural knowledge with respect to dedicated tools for architectural knowledge management.

In [5], authors describe their experiences in using semantic wikis for architectural knowledge management. The semantic wikis used are based on Semantic MediaWiki and OntoWiki[5]. A common feature of this work and our approach is that both support the integration of wiki-based architectural information with structured architecture models, however in [5] the semantic wiki is integrated only with the ArchiMate architectural language, while our approach is agnostic to the architectural language being used.

Summing up, the main novel aspects of our approach with respect to related work can be summarized as (i) the

[2]Implementation of the approach: http://goo.gl/mVZsT4

[3]http://www.mediawiki.org

[4]http://semantic-mediawiki.org

[5]http://aksw.org/Projects/OntoWiki.html

ability to keep structured architecture models (rather than plain images, for example) continuously synchronized with wiki-based documentation (even after modification of the models or the wiki pages); (ii) the independence of the proposed approach with respect to the architectural language used for architecture description; and (iii) the usage of well-established technologies like wikis and MDE (reducing the learning curve for our users). A more complete description of the strengths and limitations of our approach is provided in Section V.

V. Discussion

A. Strengths of the Approach

The proposed approach allows architects to achieve *continual synchronisation of wiki-based documentation and model-driven engineering artifacts*. It is important to highlight two important aspects of our approach: (i) once the mapping between the architecture language metamodel and the semantic wiki ontology is established, the "‘wiki-to-models’" synchronisation is *automatic* and requires no user interaction to be maintained, and (ii) our engine is able to preserve the synchronisation in a *continuous* manner by triggering the synchonisation procedure every time a relevant portion of either the semantic wiki or a modelling artifact is changed.

In contrast to most other research reported in recent years (see Section IV), the proposed approach is *totally agnostic to the architectural language* used for representing the software architecture of the system and to the kind of project being developed. Indeed, once the mapping between the architecture language metamodel and the ontology underlying the semantic wiki is established, that mapping is automatically preserved by our engine when dealing with the wiki-based documentation and the modeling artifacts. It is important to note that the initial mapping between the architecture language metamodel and the semantic wiki ontology is automatically generated too.

Our approach is *process independent*. That is, it does not force software architects to follow a specific and potential unfamiliar development process. In particular, the proposed approach focuses on the continuous alignment of wiki-based documentation and architectural models, and is independent of the other activities involved in the system life-cycle (e.g., requirements, implementation, testing, etc.).

B. Current Limitations of the Approach

We assume that *no concurrent modifications* will be performed to a wiki page and the corresponding portion of an architectural model. We are aware that this assumption is quite limiting, but we introduced it in order to keep the complexity of the proposed solution manageable. One of the top priorities of our future work is to extend our approach for supporting concurrent modifications.

Currently, in each software project *only a single architecture language can be used* for the models being synchronised by our engine. Again, we accepted this limitation in order to reduce the complexity of the solution. At this stage of our work we judged that this limitation is acceptable, since in our recent survey [14] about the usage of architecure languages in industry, it emerged that about 79% of software architects do not use multiple ALs within the same project.

The proposed approach synchronises semantic wiki pages and architectural models, but it is *not integrated with analysis engines* that can be used to process those models. It may be interesting to investigate the possibility of performing analyses of architectural models directly from their corresponding wiki pages and, more interestingly, on managing the feedback and results of those analyses in the semantic wiki.

VI. Conclusions and Future Work

This work proposes an approach for representing architectural information in terms of wiki-pages and MDE models; these two technical spaces are kept continuously aligned, so that they can be used interchangeably, depending on the nature of the activity being carried out.

We aim to continue our work in this direction in several ways. Firstly, we will work to deal with the current limitations discussed in Section V. Then, a thorough experimental validation will be carried out, both in terms of application to real case studies, and subsequent evaluation of the opinions of practicing software achitects as to the usefulness and practicality of the tool (and the underlying approach). In the longer term, we aim to explore the use of sketch-based approaches with the aim of automatically building and synchronizing sketch-based approaches with MDE-based models.

References

[1] M. A. Babar and I. Gorton. A tool for managing software architecture knowledge. In *Shark*, SHARK-ADI '07, pages 11–, Washington, DC, USA, 2007. IEEE Computer Society.

[2] A. Bertolino, P. Inverardi, and H. Muccini. Software architecture-based analysis and testing: a look into achievements and future challenges. *Computing*, 95(8):633–648, 2013.

[3] R. Capilla, F. Nava, S. Pérez, and J. C. Dueñas. A web-based tool for managing architectural design decisions. *SIGSOFT SEN*, 31(5), Sept. 2006.

[4] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2nd edition, 2010.

[5] R. de Boer and H. van Vliet. Experiences with semantic wikis for architectural knowledge management. In *Software Architecture, 9th Working IEEE/IFIP Conference on*, pages 32–41, 2011.

[6] D. Di Ruscio, I. Malavolta, H. Muccini, P. Pelliccione, and A. Pierantonio. Model-driven techniques to enhance architectural languages interoperability. In *Fundamental Approaches to Software Engineering, FASE*, pages 26–42. Springer Berlin/Heidelberg, 2012.

[7] G. H. Fairbanks. *Just Enough Software Architecture: A Risk-Driven Approach*. Marshall and Brainerd, 1st edition, August 2010.

[8] R. Farenhorst and H. van Vliet. Experiences with a wiki to support architectural knowledge sharing. In *Wiki4SE, Porto, Portugal*, 2008.

[9] J.-M. Favre. Cacophony: metamodel-driven software architecture reconstruction. In *Reverse Engineering, 2004. Proceedings. 11th Working Conference on*, pages 204–213, 2004.

[10] P. Feiler and D. Gluch. *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*. SEI Series in Software Engineering. Pearson Education, 2012.

[11] R. Hilliard, I. Malavolta, H. Muccini, and P. Pelliccione. On the composition and reuse of viewpoints across architecture frameworks. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, pages 131–140. IEEE, 2012.

[12] ISO. *ISO/IEC/IEEE 42010, Systems and software engineering - Architecture description*, Dec. 2011.

[13] S. Kent. Model driven engineering. In *Proceedings of the Third International Conference on Integrated Formal Methods*, IFM '02, pages 286–298, London, UK, UK, 2002. Springer-Verlag.

[14] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang. What industry needs from architectural languages: A survey. *IEEE Transactions on Software Engineering*, 39(6):869–891, 2013.

[15] F. S. Parreiras, S. Staab, and A. Winter. On marrying ontological and metamodeling technical spaces. In *ESEC/FSE, 439–448*. ACM, 2007.

[16] S. Schaffert, F. Bry, J. Baumeister, and M. Kiesel. Semantic wikis. *software, IEEE*, 25(4):8–11, 2008.