Realizing architecture frameworks through megamodelling techniques*

Rich Hilliard[1], Ivano Malavolta[2], Henry Muccini[2], Patrizio Pelliccione[2]
[1]Consulting software systems architect
r.hilliard@computer.org
[2]Dipartimento di Informatica, Università dell'Aquila, L'Aquila - Italy
{ivano.malavolta,henry.muccini,patrizio.pelliccione}@univaq.it

ABSTRACT

Most practising software architects operate within an *architecture framework* which is a coordinated set of viewpoints, models and notations prescribed for them. Whereas architecture frameworks are defined to varying degrees of rigour and offer varying levels of tool support, they tend to be *closed*: constituent elements are defined in different non-standard ways, they are not re-usable, and the creation of other frameworks requires a complete rework.

With the aim to manage this issue, this paper presents MEGAF, an infrastructure for realizing architecture frameworks, which can be used to create architecture descriptions. It builds upon the conceptual foundations of ISO/IEC 42010 for architecture description. MEGAF is realized through megamodeling techniques and is implemented via Eclipse plugins.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*; D.2.11 [**Software Engineering**]: Software Architectures; D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

Design, Modeling.

Keywords

Software Architecture, ADL, model driven, metamodelling, megamodelling.

1. INTRODUCTION

From the earliest work in Software Architecture (SA), it has been a fundamental tenet of the field that architectures are best expressed in terms of *multiple views* [12]. Each architecture view depicts some aspects, or *system concerns*, to address the needs of various stakeholders, while other views address other concerns; taken together, these views yield a picture of the architecture as a whole.

*This work is partly supported by the Italian PRIN d-ASAP project.

The use of multiple views has become standard practice in industry [10, 6, 3, 7]. Academic research and existing architecture description languages have focused predominantly on the structural view (i.e. components and connectors) and sometimes on behaviour at the architectural level. They have offered limited support to address the needs of stakeholders with different concerns such as data management, safety, security, reliability and so on[1].

One consequence of the tenet of using multiple views is a growing body of viewpoints that have become available, such as [10, 13, 9]. A second consequence is the rise of *architecture frameworks* as coordinated sets of viewpoints. Most practising software architects must operate within an architecture framework prescribed for them by their organization or client. Current frameworks tend to be *closed*—as a result, (i) it is difficult to re-use viewpoints and concerns for defining new frameworks to be used in different organizations or domains; and (ii) it is impossible to define consistency rules among viewpoints once forever, since such rules are not reusable as the main artefacts themselves.

With the aim of taking a step towards the solution of these current limitations, the *goal of this paper* is to provide an infrastructure, called MEGAF, for building reusable architecture frameworks.

As analysed in [5], once an organization has defined a framework to be used within its domain, the organization can more easily capitalize investments in evaluation, training, and automated tools. Using the conceptual foundations of ISO/IEC 42010 [8], we show how automated support for viewpoints, as first-class entities, can provide architects with improved support for architectural modelling and its application to the analysis of architecture descriptions. Our approach also provides a basis for sharing and reuse of such capabilities across projects (and the community) as an interface to a repository of reusable architectural knowledge. More precisely, MEGAF allows software architects to create new architecture frameworks by means of the following features: i) it provides mechanisms to *store* views, viewpoints, stakeholders and system concerns; ii) it provides mechanisms to *define correspondences* among views, viewpoints, stakeholders, system concerns and even among architectural elements that are part of them; iii) it enables *consistency and completeness* checks based on defined architectural relationships and rules among elements.

MEGAF is realized by means of megamodeling techniques [1] that provide appropriate ways for handling different type of models. By considering views, viewpoints, stakeholders, and concerns as first class elements of a megamodel, MEGAF allows software architects to define, store, and combine them in order to generate the desired architecture framework. In order to understand the power of this technology, by representing views as sets of models we are

[1]See [8] for an extensive discussion of the range of architecturally-relevant system concerns.

able, for instance, to define links among different views, to define links between artefacts that constitute each view, and to create and store views in libraries.

The paper is structured as follows: Section 2 introduces architecture frameworks, the conceptual foundations of ISO/IEC 42010 for architecture description, and highlights the main existing problems that represent the motivation of this paper. Section 3 is the heart of this paper and presents our approach for realizing architecture frameworks. Section 4 concludes the paper by highlighting future research directions.

2. ARCHITECTURE DESCRIPTIONS AND ARCHITECTURE FRAMEWORKS

ISO/IEC 42010, *Software and System Engineering — Architecture Description* [8], is the internationalized version of IEEE Std 1471, first published in 2000 [7]. The standard addresses *architecture description*: the practices of recording software, system and enterprise architectures in a consistent form so that they may be understood, documented, analysed and realized. The standard is method-neutral; designed to be usable by architects employing many different architecting methods. ISO/IEC 42010 brings into focus mechanisms for reuse and interoperability of architecting techniques through three mechanisms: (i) *architecture viewpoints*: common ways of expressing and solving a set of known architectural concerns that may be reused across projects; (ii) *architecture frameworks*: coordinated set of viewpoints for use by a particular stakeholder community or domain of application; (iii) *architecture description languages* (ADLs) [11] capable of expressing certain system concerns through one or more modelling resources.

Architecture viewpoints, as defined by the standard, codify the practice in architecting of specifying an architecture via multiple views of that architecture where each view is created using some set of conventions, notations and modelling practices. These conventions and associated practices form the viewpoint. The key idea of an architecture viewpoint is *a set of modelling resources able to address a particular set of system concerns for a particular audience of system stakeholders*. As such, a viewpoint is a form of reusable architectural knowledge (like a pattern or style) for solving a certain kind of architectural description problem with tried-and-tested modelling techniques.

An *architecture framework* builds on the viewpoint idea as a coordinated set of viewpoints, conventions, principles and practices of architecture description established within a specific domain of application or community of stakeholders. Similarly, an *ADL* is a packaging of one or more types of model (usually unified by a common syntax and semantics) enabling certain system concerns to be expressed through one or more types of modelling. For a discussion of the proposed content model and mechanism for architecture frameworks in ISO/IEC 42010, see [5].

The idea of an *architecture framework* dates back to the 1970s. In enterprise architecture, Zachman popularized the term through his information systems architecture framework [14]. Since then, many frameworks have been proposed, published and used, in a variety of domains and defined with varying degrees of formality. Recent frameworks include GERAM, TOGAF, and DODAF. Architecting methods are often presented as frameworks, i.e. as a coordinated set of viewpoints to use [10, 6, 13, 2, 4]. The recurring idea among these is that an architecture framework is a prefabricated structure that one can use to organize an architecture description into complementary views [5].

The architecture frameworks proposed till now have been defined with limited degrees of rigour; limiting users' ability to reuse

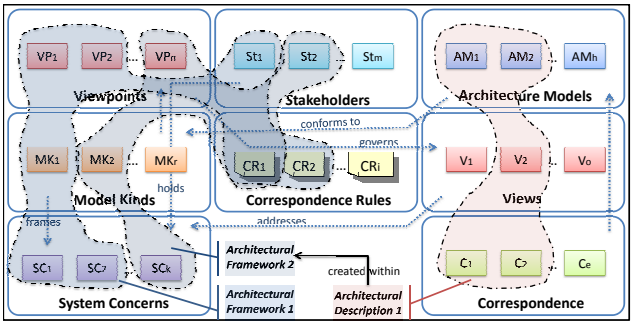

Figure 1: Overview of MEGAF

those frameworks and their constituent elements. Furthermore, while there is a growing number of notations and automated tools to support them, solutions to the general problems of architecture description are still scattered. Much work is done with notations (such as UML) not directly suited to specialized system concerns (such as performance, availability or cost); most automated tools are closed—supporting at most one architecture framework or only a few types of notations; the tools for cross-model or cross-view checking are limited or do not exist.

In the remainder of this paper we demonstrate how to *reify* the ontology of ISO/IEC 42010 to provide automated support for a number of features that will address part of the limitations above discussed.

3. MEGAMODELLING ARCHITECTURE FRAMEWORKS

Megamodeling has been proposed with the aim of supporting modelling in the large, i.e. dealing with models, metamodels, and their properties and relations. Intuitively, a megamodel is a model of which at least some elements represent and/or refer to models or metamodels. Megamodeling offers the possibility to specify semantic links between models (and metamodels) and to navigate among them. This is fundamental in MEGAF since architecture views generally have important relations defined among them.

In order to have a homogeneous framework we make the assumption that all of the heterogeneous artefacts that we use are models and that each model conforms to its metamodel. This enables the management of complex artefacts since their complexity is defined and encoded in the metamodel, thus enabling programmatic management of (even complex) models. This assumption follows a basic tenet of the ISO/IEC 42010 standard: each view conforms to a viewpoint, and each architecture model conforms to a model kind. The assumption also seems reasonable in light of the recent Doc2Model (Document to Model[2]) Eclipse project for parsing structured documents to produce EMF[3] models.

The result is MEGAF, which is an infrastructure for realizing architecture frameworks which can be used to create architecture descriptions. By referring to Figure 1:

▶ MEGAF is an extensible repository of viewpoints, views, model kinds, architecture models, system concerns, and stakeholders.

▶ correspondences and correspondence rules between arbitrary elements can be created in MEGAF. They enable the architect to express and enforce relations both between various elements inside an architecture description and *across* architecture descriptions (such as for product lines or systems of systems).

▶ MEGAF provides functionalities that allow software architects to

[2] http://eclipse.org/proposals/doc2model

[3] http://www.eclipse.org/modeling/emf

create their own framework by properly selecting among artefacts previously defined and resident inside MEGAF.

► Once the framework has been defined, it can be used to realize the architecture description of the system-of-interest. MEGAF allows the architecture description to be created compliant to the architecture framework, such as the realized architecture models conform to suitable model kinds contained in the architecture framework; moreover views are governed by viewpoints defined in the architecture framework and address some system concerns.

We identified three potential classes of users that would benefit from this work:

► "ordinary" architects (and their organizations) are most likely to use a predefined architecture framework. These end-users do not create new viewpoints, but use an existing framework "out of the box", however, they may need to customize the presentation of architecture descriptions for various stakeholder audiences or integrate that framework with existing project tools or artefacts.

► "senior" architects create new viewpoints or model kinds to address particular system concerns (e.g. fault tolerance) not addressed by a current framework. They would then integrate those by linking them with other existing representations.

► researchers develop new analysis techniques or representations on top of existing base derived from the previous two cases.

In the remainder of the paper we make use of a running example to show how an architecture description and an architecture framework may be modelled using our approach. The running example is a simple architecture description for a Subscription-based Sensor Collection Service (SBSCS)[4].

3.1 Conceptual overview of MEGAF

The initial step for building MEGAF is the creation of a generic metamodel for software architecture megamodels; we call this metamodel GMM4SA that stands for Global Model Management for Software Architectures. Models conforming to GMM4SA have other models as first class entities, such as architectural models, their metamodels, external models for representing correspondences, and so on. Relationships expressed in the ISO/IEC 42010 standard have been encoded into GMM4SA, so that each megamodel conforming to it must satisfy those relationships in order to be valid. Additional relationships and rules, specific to an architecture framework or organization are defined by means of weaving models and OCL[5] constraints; these implement correspondences and correspondence rules, respectively. Therefore, each architecture description (AD) element (e.g. view, viewpoint, stakeholder, concern, etc.) can be explicitly represented via a model or a model element depending on its granularity (e.g. the *SystemConcern* becomes a metaclass that we can instantiate as many times we need to model different concerns, whereas *ArchitectureModel* becomes a metaclass that references an external resource, so its instances are externally defined models). The interested reader can find the GMM4SA metamodel on the web-page of MEGAF.

Referring to the three features of MEGAF highlighted in the introduction, a megamodel conforming to GMM4SA can be considered a container in which views, viewpoints, stakeholders and system concerns are stored and maintained (point (i)). As described before, GMM4SA offers two mechanisms to define correspondences among AD elements (point (ii)): the first is *Correspondence* that allows software architects to define weaving relationships between AD elements; the second is *CorrespondenceRules* expressed in terms of OCL constraints. These constraints can be defined between metamodels, models, or AD elements (point (iii)).

Correspondences among AD elements can be used by software architects to perform strong and powerful consistency checks that can be defined at varying levels of granularity: (a) conformance with respect to the standard and rules defined therein, (b) checks of rules that are framework-specific, (c) checks of rules that are "local" to the specific architecture description. While (a)-rules should be defined once and forever, (b)-rules allow software architects to properly define or constrain a framework, and (c)-rules allow software architects to properly realize the software architecture description of the system-of-interest.

Figure 2 shows an overview of a megamodel defined for the running example SBSCS that conforms to GMM4SA. The megamodel is conceptually divided into two parts, the architecture framework and the architecture description. The architecture framework contains three viewpoints, namely the financial, the operational and the system viewpoints. Each viewpoint governs a view that is referenced in the architecture description. Associated to each viewpoint there is a model kind, as shown in the figure. Model kinds are described by a set of common information like overview, description and references to external documents. Moreover, each model kind description is completed by its underlying metamodel. Architecture models conforming to these metamodels are part of the architecture description. As can be seen in figure, the *SCS Dataflow* architecture model is realized with a graphical editor that we built for the SID model kind. The *Collection TLD* architecture model is realized by using the default tree-like editor. Finally, the *SCP profit statement* architecture model is realized by means of an excel sheet, since we instrumented MEGAF with ATL transformations able to import/export excel files.

In this simple example each viewpoint frames a system concern and each system concern is addressed by a view. Finally, the architecture description identifies only three of the eight stakeholders associated to the framework; this is because the other stakeholders do not have any concern related to the SBSCS architecture being described. In the following section we explain the implementation aspects of MEGAF by making use of this running example.

3.2 Implementation of MEGAF

The current MEGAF prototype is available at the MEGAF webpage[6]. MEGAF is implemented as an Eclipse plugin and more specifically, it is defined as an extension of the AM3 component of AMMA[7]. This extension is composed of three main elements: (i) the GMM4SA metamegamodel, (ii) a set of Java classes that extend the core AM3 viewers and implement some auxiliary mechanisms of MEGAF, and (iii) a set of OCL and ATL specifications implementing the consistency and completeness checks for megamodels conforming to GMM4SA.

Some functionalities of MEGAF are inherited from the AM3 engine and from its available extensions. In order to provide software architecture-specific functionalities we needed to extend the core AM3 plugins in the context of the Eclipse and AMMA platforms.

First of all, due to its generality, the EMF editor is very basic; then we extend it for specifying attributes and references related to GMM4SA only. For example, we provide an editor for architecture viewpoints in which the software architect can specify which system concerns are framed by the current viewpoint; or for each architecture description element, the software architect can specify which architecture decisions affect it, and so on.

[4] http://www.iso-architecture.org/ieee-1471/docs/SBSCS-AD-v02.pdf

[5] OMG Object Constraint Language (OCL): http://www.omg.org/spec/OCL

[6] MEGAF website: http://megaf.di.univaq.it

[7] AMMA website: http://www.sciences.univ-nantes.fr/lina/atl/AMMAROOT

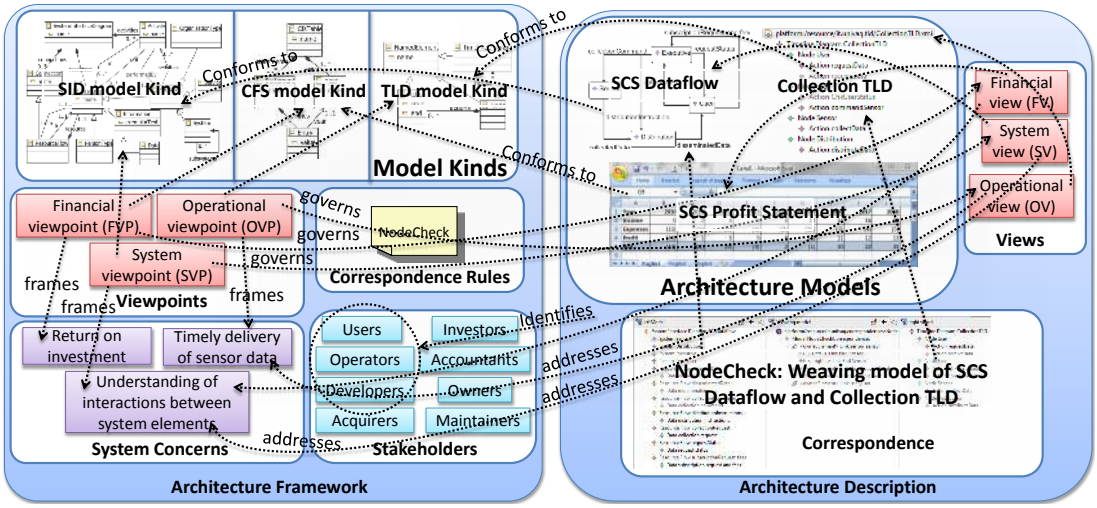

Figure 2: Megamodel for the running example SBSCS

As conceptually highlighted in Section 3.1, we defined a number of OCL constraints to help software architects in checking various properties of the current architecture description. These OCL constraints are implemented as ATL queries since the whole MEGAF tooling set is based on the AMMA platform (ATL is part of it as well) and ATL provides a stable and intuitive implementation of the OCL language; further on, ATL queries can be launched either programmatically or via Ant scripts[8]. This results in a homogeneous framework to orchestrate, manage and configure such queries within MEGAF.

The MEGAF tooling set can be extended: we provide an extension point to define how to import specific kinds of models into a megamodel conforming to GMM4SA. For instance, a set of stakeholders and concerns may be extracted from a stakeholder diagram, or a set of decisions and rationales may be extracted from some other Architecture Knowledge diagram, a specific model may be extracted from an Excel sheet, and so on. In the SBSCS running example, we implemented an extension of MEGAF that is able to analyse a financial model in Excel and produce its corresponding EMF model; this allows us to specify financial properties using Excel and consider this information in the megamodel of the SBSCS architecture description.

4. CONCLUSION AND FUTURE WORK

This paper proposed MEGAF that is an infrastructure for creating architecture frameworks that can be used for realizing architecture descriptions. MEGAF is realized via megamodeling techniques that natively promote the reuse of each architectural element that resides in MEGAF: a framework can be created by simply linking and reusing existing elements or adding new ones if needed. MEGAF and its features have been presented by means of a simple running example.

On the future work side, we plan to investigate more powerful mechanisms to support the reuse, from the reuse of a framework to the reuse of a single system concern. At the moment an artefact can be reused as it is. Modifications can be made by hand, possibly starting from a copy of the existing element. We are currently investigating automatic extension and customization mechanisms inspired by the work presented in [3], which, by means of composition operators, enables the customization and extension of ADLs.

[8] `http://ant.apache.org`

5. REFERENCES

[1] J. Bézivin, F. Jouault, P. Rosenthal, and P. Valduriez. Modeling in the large and modeling in the small. In *LNCS, Vol. 3599*, 2005.

[2] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2003.

[3] D. Di Ruscio, I. Malavolta, H. Muccini, P. Pelliccione, and A. Pierantonio. Developing next generation ADLs through MDE techniques. In *ICSE 2010*, 2010.

[4] P. Eeles and P. Cripps. *The Process of Software Architecting*. Addison Wesley, 2010.

[5] D. Emery and R. Hilliard. Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. In *WICSA/ECSA 2009*, 2009.

[6] C. Hofmeister, R. L. Nord, and D. Soni. *Applied Software Architecture*. Addison-Wesley, 2000.

[7] IEEE. *IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, October 2000.

[8] ISO. *ISO/IEC CD1 42010, Systems and software engineering — Architecture description (draft)*, January 2010.

[9] P. Kruchten, R. Capilla, and J. C. Dueñas. The decision view's role in software architecture practice. *IEEE Software*, 26(2):36–42, 2009.

[10] P. B. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6), 1995.

[11] N. Medvidovic and R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE TSE*, 26(1), 2000.

[12] D. E. Perry and A. L. Wolf. Foundations for the Study of Software Architecture. *SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[13] N. Rozanski and E. Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2005.

[14] J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.