Extending ANSI/IEEE Standard 1471 for Representing Architectural Rationale

Simon Giesecke and Jasminka Matevska and Wilhelm Hasselbring

Carl von Ossietzky University of Oldenburg, Software Engineering Group,
26111 Oldenburg (Oldb.), Germany,
{giesecke,matevska,hasselbring}@informatik.uni-oldenburg.de

Abstract. Both software engineering research community and standardisation organisations recognized a need for a general standard as a guideline for modelling software architectures. Even though the ANSI/-IEEE Standard 1471 provides a widely applicable reference model for that purpose, it merely identifies a need of a rationale for the architectural concepts selected, but does not include any further principles. We present one possible extension of the standard which integrates the concepts of architectural style, and identifies rationale on the model, view and viewpoint levels.

1 Introduction and Motivation

The motivation for the work can be found in the context of style-based modelling of software architectures. Many definitions of architectural styles and patterns exist, as well as definitions of similar concepts that play a role in the description of software architecture. Most of these definitions are vague and even conflicting [1]. Therefore, we see the need to relate these concepts to an established reference model for architectural descriptions. The ANSI/IEEE Standard 1471 provides the most elaborate, widely applicable reference model for our purpose. We extend the work described in [1] by relating the identified concepts to the elements already existing in the standard.

Furthermore, there is a possibility of mapping specific the UML onto the reference model. For example, a viewpoint may prescribe the use of certain UML diagram types. UML 2.0 introduces *collaborations* as a notation part of *Composite Structure Diagrams*. These may be used to model particular viewpoints and even describing architectural patterns.

2 Foundations and State of the Art

ANSI/IEEE Standard 1471 The ANSI/IEEE Standard 1471 provides a well-founded conceptual reference model for software architecture as well as a definition of software architecture. The UML class model shown in Figure 1 is directly reproduced from the standard. However, the standard itself notes that this model is not authoritative with respect to the relationships of the represented concepts, but should merely serve as an illustration of the standard's intentions.

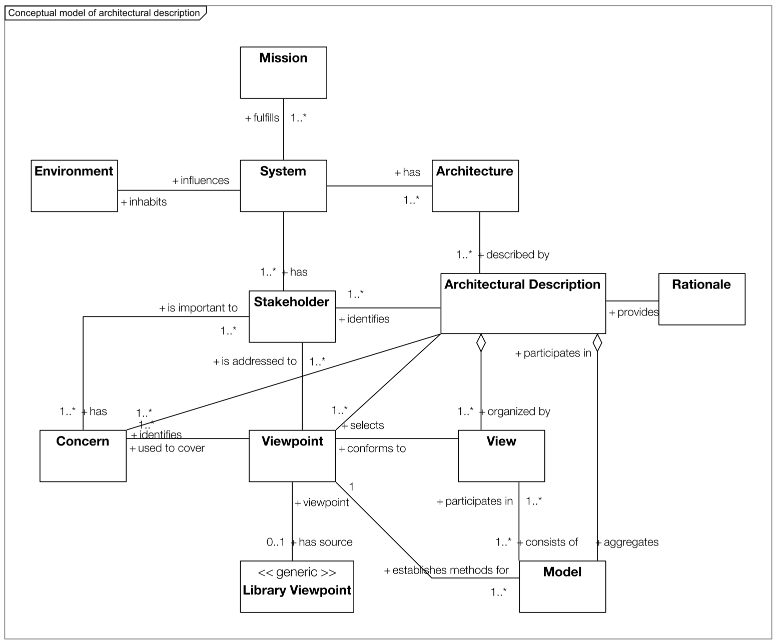

Fig. 1. ANSI/IEEE 1471 Reference Model

Generic elements An exception to this general rule is the Library Viewpoint concept, since it explicitly refers to entities that are not confined to a single architectural description, but are reused across projects. This leads us to introduce a deviation of the original reference model and categorise the class with the generic stereotype. We will use this stereotype in our extension as well.

Description elements The standard does not explicitly make assumptions on the form in which the architectural description is represented, other than that the primary information is contained in the models (rather than the views, e.g., which are only conceptual entities that organise information contained in the models). It remains unspecified which description elements other than the models form part of an architectural description. In other words, it is unclear how to map the parts of a concrete architectural description to the conceptual elements provided by the reference model. While this is possible for the single architectural models, it is impossible for description elements that are not constrained to a single model: for example, the metainformation on which viewpoints have been chosen or which models are present are not part of any model, but have to be found elsewhere. It is not the intention of a standard to provide some logical or physical structure for an architecture description, but it should still be possible to use the vocabulary defined by the standard to refer to the elements of such structures defined by third parties.

In the original standard, architectural rationale is defined to comprise "the rationale for the architectural concepts selected" and "evidence of the consideration of alternative architectural concepts and the rationale for the choices made".

Conclusion With these restrictions in mind, we still deem the standard's reference model useful to provide a basis for presenting our extensions. It is important to note that the purpose of the reference model is not to provide a meta-model for instantiating architectural descriptions, neither manually nor tool-supported, but to provide the vocabulary necessary for communicating about architectural description concepts and for defining specific architectural description methods.

Related Work In [2], another extension of the IEEE Standard 1471 reference model for representing architectural rationale is proposed. They introduce the additional classes ArchitecturePrinciple, ArchitectureDecision, QoSConstraint, ArchitectureOption and Eval[uation]Criteria. ArchitectureDecisions are linked to the views they influence, while ArchitecturePrinciples are derived from the Mission.

3 Proposed Standard Extension

In accordance with the rationale underlying the standard itself, we propose a lightweight extension of the standard, which ought to be general enough to be applicable to any architectural description practice that can be mapped onto the original standard. We introduce rationale documentation for each of the central concepts viewpoints, views and models. While that view is not explicitly held in the standard, we assume these three concepts can be regarded as three layers of architectural description in the stated order.

In an analogous hierarchy, architectural rationale can be structured along the same layers:

Rationale for Viewpoint Selection As the standard already notes, part of the architectural rationale is the documentation of the reasons for choosing the set of viewpoints used.

The standard already introduces the notion of a library viewpoint, which is a viewpoint that is typically used in documenting software architectures in a particular application domain, for example. However, even more useful are collections of viewpoints that often occur together and provide a meaningful coverage of typically relevant architectural concerns. One example for such a set of viewpoints are those defined in the Reference Model for Open Distributed Processing (RM-ODP) (ISO/IEC Standard 10746-1).

Rationale for View Principles Next, on the level of views, the principles for organizing the contents of the view must be determined. Important means, especially for structurally oriented views, are *architectural styles.* Analogously to the distinction of Architecture and ArchitecturalDescription made by the standard, we distinguish a Style (as a conceptual entity) from a StyleDescription (a representation of the former conceptual entity). The question whether multiple styles can be combined or if they may overlap within a model remains unresolved [3], so we do not explicitly introduce a restriction in this direction into our model. When applying the reference model, it must be decided if such a restriction should be made or not.

Styles may be taken from a library associated with the viewpoint, i.e. a viewpoint *proposes* certain styles, and thus be intended to be reused across different projects. Alternatively, it may be specifically modelled for the project at hand.

A style establishes principles for modeling within the view, but does not yet instantiate any concrete elements, e.g. the choice for a pipe-and-filter style does not instantiate any concrete pipe or filter, which is a separate design activity. This is different than for design patterns: E.g., when choosing the Singleton pattern [4], the choice of the Singleton pattern and the identification of the class which should be made a singleton cannot be separated but form a single atomic design decision. However, for a style, patterns may be defined that can be instantiated once the style has been selected.

The view principles rationale may also select a ReferenceArchitecture which serves as the basis for this architectural view.

Rationale for Model Elements Concrete elements representing system entities are represented in the models of an architectural description. The documentation of reasons for choosing certain elements over others are thus documented on this rationale level. This documentation cannot always be associated with single elements of a model, sometimes multiple elements must be seen in context, for example when instantiating patterns associated with a style [5].

Each of the architectural rationale parts is thought to be a description element on its own. They thus accompany the primary description elements (the models). We explicitly introduce the elements of the architectural rationale as description elements that convey information on their own.

Justification It might seem natural to model ReferenceArchitecture as a specialisation of architecture. However, we decided against this for two reasons: First, specialisation relationships are not considered at all in the original standard, and so the introduction for this class would impair the coherence of the reference model. Second, an architecture is associated with a single system, which is obviously not the case for a reference architecture, which make only sense if it is applied to multiple systems. Adapting the standard to restrain from this problem would require several further changes to the standard, which we chose not to make.

As indicated above, we associate a StyleDescription with a View. This is a deliberate design choice, and its alternatives were to associate a style with either a Model or with the ArchitecturalDescription as a whole. We decided not to associate a style with a model, because a style governs a decomposi-

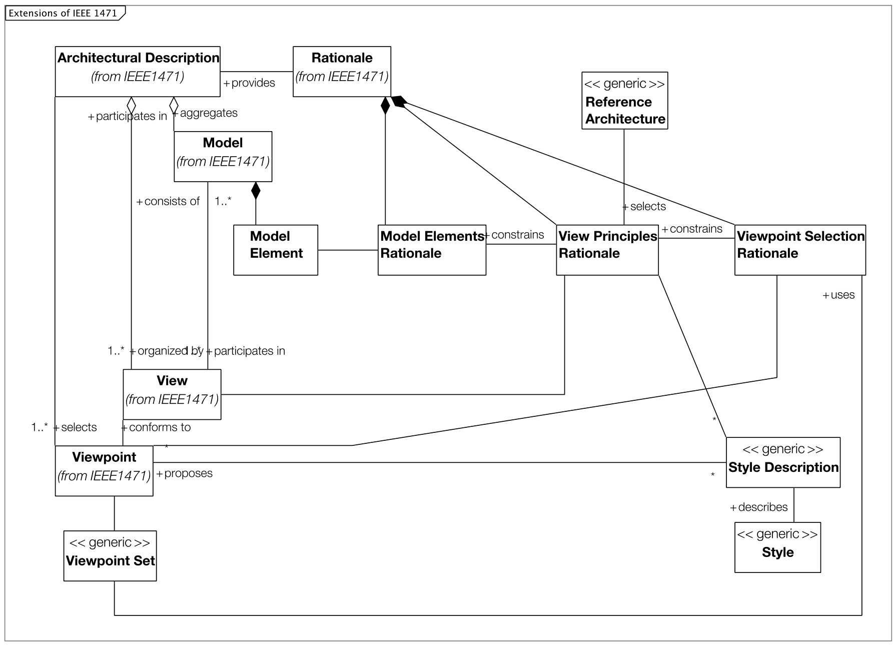

Fig. 2. Extension of the Reference Model

tion of the system which may be referenced in multiple models. Furthermore, we decided against associating a style with the architectural description as a whole, since different decompositions of a system (e.g. its run-time decomposition into process components vs. its design-time decomposition into module components) may be governed by different styles with no immediate relationship.

References

1. Giesecke, S., Hasselbring, W., Riebisch, M.: Classifying architectural constraints as a basis for software quality assessment. Advanced Engineering Informatics (2006) Accepted for publication.
2. Sarkar, S., Thonse, S.: EAML – architecture modeling language for enterprise applications. In: CEC-EAST '04: Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference on (CEC-East'04), Washington, DC, USA, IEEE Computer Society (2004) 40–47
3. Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., Little, R.: Documenting Software Architectures: Views and Beyond. Pearson Education (2002)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley (1995)
5. Garlan, D.: What is style? In Garlan, D., ed.: Software architectures. Volume 106 of Dagstuhl-Seminar-Report., Saarbrücken, Germany (1995) Proceedings of the Dagstuhl Workshop on Software Architecture.