

View Composition as a First-Class Concept in Architectural Descriptions

Nelis Boucké, Danny Weyns and Tom Holvoet

Distrinet, Departement of Computerscience, KULeuven

Celestijnenlaan 200A, 3001, Leuven, Belgium.

Email: {nelis.boucke,danny.weyns,tom.holvoet}@cs.kuleuven.be

Abstract—Our position is that architectural descriptions lack composition of views, preventing a proper separation of concerns. This position took shape from experiences with building several complex distributed software systems. Our claim is that view composition should be a first-class entity in architectural descriptions. As a first step, we propose an extension of the conceptual model of IEEE Recommended Practice for Architectural Description of Software-Intensive Systems with view composition.

I. CONTEXT

Our research group performs application driven research to built advanced open and distributed software systems, with as main challenges a high degree of dynamism and change in either the problem or the system’s environment together with a physically distributed infrastructure. More specifically, we use a multiagent architecture for modeling such systems. In a multiagent architecture, the system is structured into a number of interacting autonomous entities (agents) embedded in an environment. Over the last five years the research of our group was guided by architecting several applications, ranging from an experimental peer-to-peer file sharing system up to an industrial automatic transportation system that uses automatic guided vehicles to transport loads in a warehouse environment.

In this paper, we focus on our experiences in specifying software architectures and place this in the context of aspect-orientation. The argumentation of this paper is mainly built around the conceptual model of IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (referred to as IEEE 1471 [17]), because it provides a concise but very clear overview of the conceptual elements and relations of in an architectural description (AD).

IEEE 1471 is centered around two key ideas: (1) a conceptual model for architectural description; and (2) a statement of what information must be included in any 1471-compliant architectural description, independent of the specific architectural language in use. The conceptual model ties together such concepts as architectural description, concerns and view.

Figure 1 shows a portion of this conceptual model in UML. An architectural description is “a collection of products to document a specific architecture”. An AD is centered around views, defined as “a representation of a whole system from the perspective of a related set of concerns”. Each view is built according to a viewpoint, defined as “a specification of the conventions for constructing and using a view; a

pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis”. Viewpoints cover specific concerns, described as “those interests which pertain to the systems development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders”. Each view exists of one or more architectural models, containing a concrete description of the systems elements and relations.

One of the strong point of modern AD with respect to aspect orientation in general, and in particular for IEEE 1471, is that it explicitly considers concerns and defines a relation between the concerns and the views covering the concerns.

Also, there is a clear relation between the separation of concerns in views and the well known vision of Multidimensional separation of concerns (MDSOC), as pointed out by Kandé et al. [18]. In this paper we follow the symmetrical vision on aspect orientation [14]. As a consequence, we do not use the term ‘aspects’ and make no distinction between ‘normal’ views and views that could be considered ‘aspectual’.

II. POSITION STATEMENT

Our position is that architectural descriptions lack composition of views, preventing a proper separation of concerns¹. View composition should be a first-class entity in the conceptual model of IEEE 1471.

The lack of support for view composition implies two important problems:

- 1) Relations between information that is documented in different views remain implicit (or are described only informally).
- 2) Although modern ADs explicitly support the notion of concern and the description of concerns in one or more views, concerns still crosscut each other in the architectural views.

The first problem leads to increased overhead to keep architectural documentation consistent, and generally leads to architectural erosion over the system’s lifetime [2].

The second problem is particularly important in the context of aspect-orientation. The IEEE 1471 conceptual model tells

¹We want to emphasize that with separating concerns in the AD, we specifically mean separating concerns in *the typical* representation of an AD, the decomposition into architectural views and models (called the dominant decomposition for an architecture in [1]).

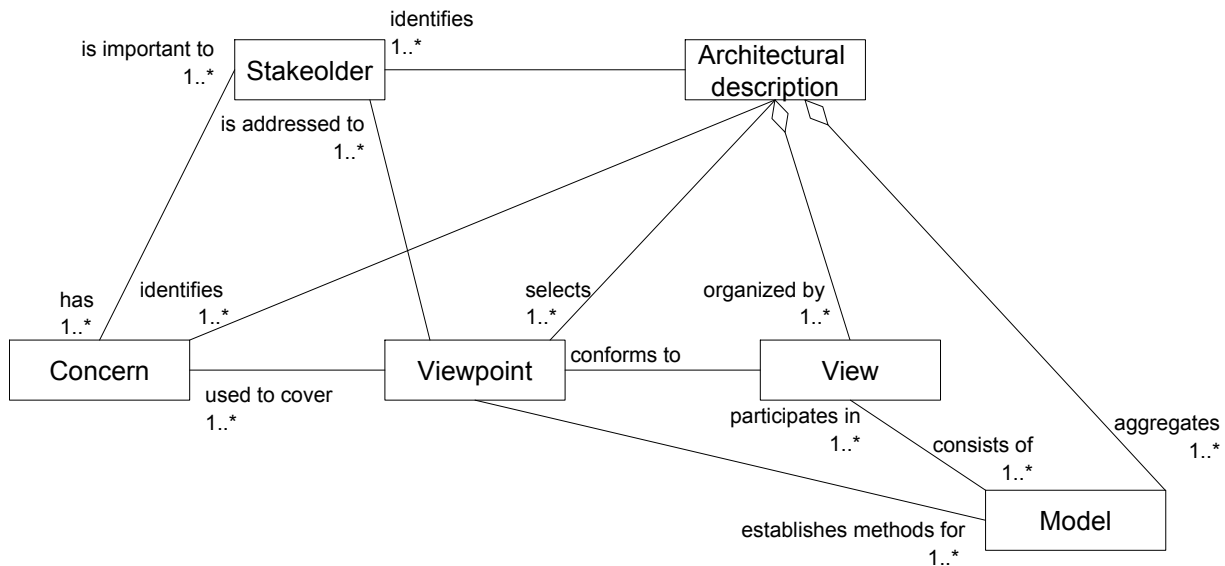


Fig. 1. A summary of important concepts in the IEEE 1471 conceptual model.

us that several views can be used to describe a single concern and that a view can contain a description covering several concerns. Although an architect can choose to limit the number of concerns described in views, in general, overlap between concerns can not be avoided [16]. As a consequence, software architect are left with no choice but to mingle concerns in several views to express relationships between concerns, or to duplicate parts of the architecture in different views. This leads to crosscutting concerns, a problem that has been extensively studied in the field of aspect-orientation.

III. INDUSTRIAL TRANSPORTATION SYSTEM

In this section, we illustrate the problems of a lack of support for view composition in an industrial Automatic Guided Vehicle Transportation System (AGVTS) that we have developed in a joint R&D project with Egemin². An AGVTS is a fully automated transportation system that uses multiple Automatic Guided Vehicles (AGVs) to transport loads in a warehouse or production plant. This system was built according to a multi-agent architecture, the architectural description can be found in [5].

For the documentation of the software architecture of the AGVTS, we used Views and Beyond (V&B [12]), an approach that is closely related to IEEE 1471 [11]. To document relations between different views, we used a mapping between views and a glossary. Whereas this information is very important to link different architectural issues with one another, the information remains intuitive and leaves important information implicitly. Moreover, revising the documentation and keeping everything up-to-date (e.g., cross-references and relations between different parts of the documentation) turned out to be especially hard and time-consuming [7].

From our experience, we also learned that important concerns are often scattered over different views in the AD. Two examples of such concerns identified when developing the AGVTS are: (1) the coordination between different agents or robots; and (2) the distribution of the system. As shown in previous work ([3], [6]) both concerns crosscut with each other in several architectural views. The problems with crosscutting concerns became particularly difficult during a revision the software architecture. Whereas in the original design the control system was developed with an agent deployed on each AGV, during the project, Egemin realized that switching towards a distributed agent-based architecture was a big step with far reaching effects for the company, not only for the software but for the whole organization. Therefore, a stepwise integration of the architecture was proposed where all agents are located on a single computer system with remote access to the low-level control software on the vehicles. In principle, the impact of such a revision would be limited since only the parts related to distribution including middleware support for distributed communication, would be affected. Unfortunately, because of the concerns crosscutting each other in the views, we had to adapt most of software architecture and change nearly every individual view of the AD. Our experiences learn that this is not an exceptional situation in an iterative, incremental development processes.

IV. EXTENDING THE CONCEPTUAL MODEL

We propose to extend the IEEE 1471 conceptual model with the concept of view composition. Figure 2 shows the extension and the relation to the remainder of the architecture. *View composition* describes the composition of two or more views. This is done by describing a number of *model relations*, each describing the relation between two or more models from different views. In this extension, an AD not only consists of

²<http://emc2.egemin.com/>

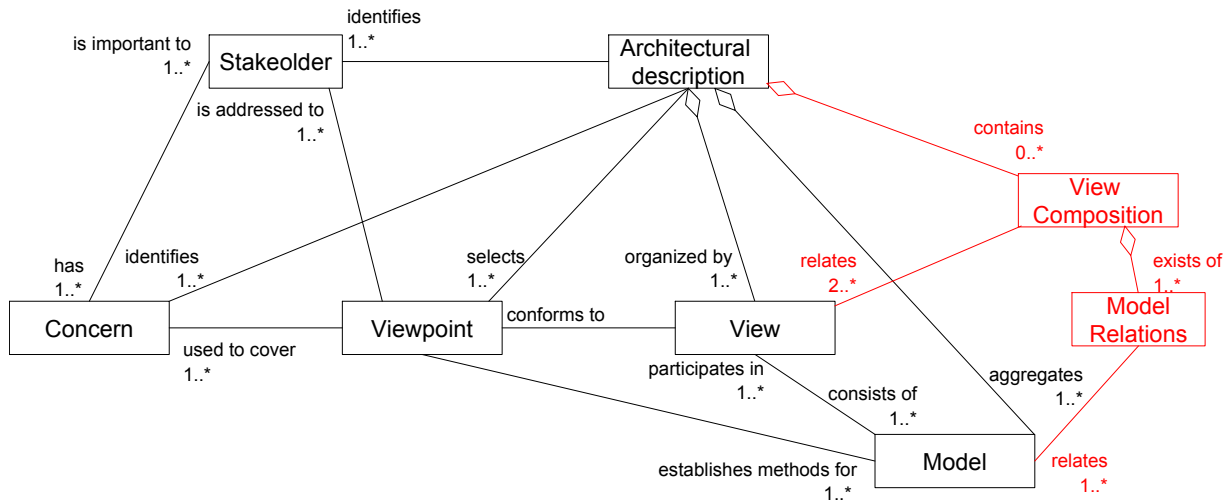


Fig. 2. An extension to the IEEE 1471 conceptual model, extensions are marked in red.

several views (containing several models) but also of a number of compositions between these views.

The extension of IEEE 1471 allows to better separate concerns and compels an architect to explicitly reason about how the different views (that describe different concerns) can be brought together to form the system.

IEEE 1471 itself is very brief on consistency between views. It points out that an AD should indicate consistency, but the recommended practice stays vague on this point. View compositions provide a explicit mechanism to prevent inconsistencies between views.

In previous work we performed several explorations of the idea of composing architectural views. [4] promotes a direct relation between concerns and view, and informally introduces operators to compose different concerns. We applied the approach in a small online auction system. As part of the AOSD-Europe report on known an new candidate concerns, we identified several important concerns in the AGVTS [19]. [6] reports our experiences with using Theme/UML for architectural design and introduces several composition operators to better support architectural composition. The operators are illustrated by describing an excerpt of the AGVTS. Finally, in [3], we introduce composition of architectural structures in the xADL language. The composition operators in this paper are rigorously defined, but limited to composing of one type of view (namely structural views describing components and connectors). Each of the extensions provides quantified [13] relations between views, an essential property of aspect-orientation.

V. DISCUSSION

The introduction of view composition in the conceptual model of IEEE 1471 is a first step, but several important questions remain open.

- A first question is how the result of a view composition (after applying all relations between models) will look. Typically, a composition results in several models and

could again be considered as a kind of view (a composite view?). Considering the result of a view composition as a view has as advantage that it in turn can be used in future composition, allowing for hierarchical composition of views. A disadvantage is that this composite view is not the same as the current definition of view in the conceptual model, since it has no associated viewpoint. Using the concept of a composite view would require an additional extension of IEEE 1471.

- The introduction of view composition leaves open how to specify the relations between models. Support for describing composition could come as specialized composition operators or as a logical language. This latter allows to reason about the views and architectural elements. View compositions could have a template definition, similar to what viewpoints are for views. More research is required on the advantages and disadvantages of different composition approaches.
- In our proposition, a view composition consists of several relations between concrete models. However, other types of relations may be possible. Examples are relations between architectural elements of different views (independently of specific models) and relations between viewpoints (that are used for the instantiation of a particular set of views). The latter introduces a relations between any views that instantiate these different viewpoints.
- Note that the composition specification can contain relations between the same type of architectural models as well as between *different types*. An example of the former is the composition of two models describing modules, an example of the latter is the composition of a model describing runtime components and a model describing processes. The relations between different types of models may be more difficult to describe.

VI. RELATED

Because of the limited space of this paper we briefly discussing three strongly related approaches.

The V&B [12] approach provides has a some concept of relations between views. The "B" part of V&B "helps the reader of the documentation to understand the relationship between views and how the architecture works as a conceptual whole". For example, documentation that applies to more than one view includes a documentation roadmap, a view template, a system overview, a mapping between views, a directory, a glossary, and a cross-view rationale. Whereas each document in the additional documentation of V&B is important, in general the information is rather informal and intuitive and leaves important information implicitly. We claim that the documentation of the relationships between views (of the same type as well as of different types) is as important as the views themselves. Therefore, relationships between views should be considered first-class in architectural description and formally specified.

In [15] Hilliard provides some initial ideas on tackling inconsistencies between views by putting forward the metaphor of views as modules. The metaphor suggest that views, like modules, address separation of concerns and that to the extend that views are not interfering, those views "encapsulate" concerns. Finally, the author points out that an appropriate definition of view interfaces could be of particular interest for this.

The ViewPoints framework [20] provides an organizational framework in which different views, and their relationships, can be explicitly represented and analyzed. Much of the research has focused on requirements engineering, but the same approach has been applied on architectural descriptions. The authors explicitly recognize the crucial need for relations between views and provide a tool to for advanced consistency checking between different view (with possible heterogeneity of representations for different models).

VII. CONCLUSION

Explicit relations or advanced compositions are a major challenge for separating concerns where we believe that aspect-orientation can contribute to architectural descriptions. Extending the architectural description with advanced composition is analogous with extending requirements with composition operators such as in [8], UML with model composition semantics such as for Theme/UML [9], [10] or enhancing the Java language with structural composition mechanisms like the ones supported by the Hyper/J programming language [21].

Yet, view composition will only come to its full extent when it is rigorously specified, integrated in concrete ADLs and supported by proper tools

ACKNOWLEDGMENT

Nelis is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

REFERENCES

- [1] E. Baniassad, P. C. Clements, J. Araujo, A. Moreira, A. Rashid, and B. Tekinerdogan. Discovering early aspects. *IEEE Software*, January/February, 2006.
- [2] T. Batista, C. Chavez, A. Garcia, C. Sant'Anna, U. Kulesza, A. Rashid, and F. Filho. Reflections on architectural connection: Seven issues on aspects and adls. In *Workshop on Early Aspects held at ICSE'06*, 2006.
- [3] N. Boucké, A. Garcia, and T. Holvoet. Composing architectural cross-cutting structures in xadl. Report CW 474, Departement of Computer Science, KULeuven, 2007. Submitted to the Early Aspect workshop on AOSD 2007.
- [4] N. Boucké and T. Holvoet. Relating architectural views with architectural concerns. In *Early Aspect workshop at ICSE*, 2006.
- [5] N. Boucké, T. Holvoet, T. Lefever, R. Sempels, K. Schelfthout, D. Weyns, and J. Wielemans. Applying the Architecture Tradeoff Analysis Method (ATAM) to an industrial multi-agent system application. Technical Report CW431, Dept. of Computer Science, KULeuven, 2005.
- [6] N. Boucké, D. Weyns, and T. Holvoet. Experiences with Theme/UML for architectural design of a multiagent system. In *Multiagent Systems and Software Architecture, Proceedings of the Special Track at Net.ObjectDays 2006*, pages 87–110. Net.ObjectDays, K.U.Leuven, 2006.
- [7] N. Boucké, D. Weyns, K. Schelfthout, and T. Holvoet. Applying the ATAM to an architecture for decentralized control of a transportation system. In *Quality of Software Architectures conference (QoSA)*, volume LNCS 4214, 2006.
- [8] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In *Aspect Oriented Software Development conf.*, 2007.
- [9] S. Clarke. Extending standard uml with model composition semantics. *Science of Computer Programming*, 44(1):71–100, 2002.
- [10] S. Clarke and E. Baniassad. *Aspect-Oriented Analysis and Design*. Addison-Wesley, 2005.
- [11] P. Clements. Comparing the sei's views and beyond approach for documenting software architectures with ansi-ieee 1471-2000. Technical Note 017, CMU/SEI, 2005.
- [12] P. Clements, F. Bachman, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures, Views and Beyond*. Addison Wesley, 2003.
- [13] R. Filman and D. Friedman. Aspect-oriented programming is quantification and obliviousness. In *Proceedings of the workshop on Advanced Separation of Concerns, OOPSLA*, 2000.
- [14] W. H. Harrison, H. L. Ossher, and P. L. Tarr. Asymmetrically vs. symmetrically organized paradigms for software composition. Technical report, IBM Research Division, published under RC22685, 2002.
- [15] R. Hilliard. "don't ask, don't tell" interfaces: Architectural views and their interfaces. In *2nd International Workshop on Living With Inconsistency*, 2001.
- [16] R. Hilliard. Ieee std 1471 and beyond (position paper). In *SEI's First Architecture Representation Workshop*, 2001.
- [17] IEEE. Recommended practice for architectural description of software-intensive systems (ansi/ieee-std-1471), September 2000.
- [18] M. M. Kande and A. Stroheier. On the role of multi-dimensional separation of concerns in software architecture. In *Proceedings OOPSLA workshop on Advanced Separation of Concerns*, 2000.
- [19] N. Loughran, A. Rashid, R. Chitchyan, N. Leidenfrost, J. Fabry, N. Cacho, A. Garcia, F. Sanen, E. Truyen, B. D. Win, W. Joosen, N. Boucké, T. Holvoet, A. Jackson, A. Nedos, N. Hatton, J. Munnely, S. Fritsch, S. Clarke, M. Amor, L. Fuentes, M. Pinto, and C. Canal. A domain analysis of key concerns known and new candidates. AOSD-Europe Deliverable D43, AOSD-Europe-KUL-6, 2006.
- [20] B. Nuseibeh, J. Kramer, and A. Finkelstein. Viewpoints: meaningful relationships are difficult! In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 676–681, Washington, DC, USA, 2003. IEEE Computer Society.
- [21] P. Tarr, H. Ossher, W. Harrison, and S. Sutton. N degrees of separation: Multi-dimensional separation of concerns. In *Int. Conf. on Software Engineering*, pages 107–119, 1999.