

View Relations and Composition, and tool support in xADL

Nelis Boucké, Danny Weyns and Tom Holvoet

Report CW 502 , October 2007



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

View Relations and Composition, and tool support in xADL

Nelis Boucké, Danny Weyns and Tom Holvoet

Report CW502 , October 2007

Department of Computer Science, K.U.Leuven

Abstract

Experience with the design of industrial strength architectures revealed a lack of support for relating views in architectural descriptions (ADs). Leaving relations among views implicit leads to ambiguity and hampers changing the AD while keeping it consistent.

We claim that relations and compositions between views should be first-class concepts of architectural descriptions languages. Recently, we introduced three concrete relations, i.e. refinement, mapping and unification. We illustrated the use of these relations in several examples using xADL. Besides improved clarity and consistency of ADs, relations between views enable (automatic) composition of views. A view composition explicitizes how two or more views and relations among these views are used for integration, resulting in an integrated view.

In this paper, we provide an overview of relations and view composition. We show how we have developed support for automatic generation of integrated views in the ArchStudio environment. We use a simplified video-on-demand system as an illustration.

Keywords : software architecture, composition, view relations and composition

CR Subject Classification : I.2.11, I.2.8, I.2.1

View Relations and Composition, and tool support in xADL

Nelis Boucké, Danny Weyns and Tom Holvoet
DistriNet, Departement of Computerscience, KULeuven
Celestijnelaan 200A, 3001 Leuven, Belgium
{nelis.boucke,danny.weyns,tom.holvoet}.cs.kuleuven.be

Abstract

Experience with the design of industrial strength architectures revealed a lack of support for relating views in architectural descriptions (ADs). Leaving relations among views implicit leads to ambiguity and hampers changing the AD while keeping it consistent.

We claim that relations and compositions between views should be first-class concepts of architectural descriptions languages. Recently, we introduced three concrete relations, i.e. refinement, mapping and unification. We illustrated the use of these relations in several examples using xADL. Besides improved clarity and consistency of ADs, relations between views enable (automatic) composition of views. A view composition explicitizes how two or more views and relations among these views are used for integration, resulting in an integrated view.

In this paper, we provide an overview of relations and view composition. We show how we have developed support for automatic generation of integrated views in the ArchStudio environment. We use a simplified video-on-demand system as an illustration.

1 Introduction

Building a software architecture comes down to defining design structures which comprise software elements, the externally visible properties of those elements, and the relationships between them [3]. Designing an architecture typically includes the selection and production of several architectural views [6, 10]. One of the strength of using views is that an architect can emphasize on specific aspect of the system. This allows to cope with complexity and divide an AD in comprehensible pieces.

Despite different emphasis, views describing the same system are typically not independent. Once an AD is divided in comprehensible pieces (views), these pieces need to be integrated to form the system as a whole. Having divided to conquer, we must reunite to rule [11].

Our experience in the development of architectures with industrial partner revealed a lack of support in current architectural practice [6, 8, 9, 12, 1] for relating views. An illustration is that the conceptual model of the IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [10] establishes terms and concepts pertaining to the content and use of architectural descriptions, but lacks concepts for relations and compositions. This lack leads to ambiguity in the architectural description and makes it difficult to change the description while keeping the architectural documentation consistent [5].

To prevent this lack, we advocate that relations and composition should become a first-class concept in an architectural description language. Recently, we introduced three concrete relations, i.e. refinement, mapping and unification. We illustrated the use of these relations in several examples using xADL. Besides improved clarity and consistency of ADs, relations between views enable (automatic) composition of views. Where unification, refinement and mapping explicitize the *relation* between two or more views, a view composition explicitizes how several views and relations are *bundled for integration*, resulting in an integrated view. As a proof of concept, we extended a representative architectural description language (ADL) by adding explicit support for composing the structural views of xADL [7] and modeled several example architectures with this extension. From this experience we learned that proper tool support is essential to make view composition feasible. For example, enabling architects to easily consult what the implications are of a relations or a change by automatically generating the integrated views.

In this paper, we provide an overview of relations and view composition. We show how we have integrated support for automatic generation of integrated views in the ArchStudio environment. Section 2 illustrates view composition and three concrete relations using an example system. Section 3 discusses tool support for automatic generation of views starting from a description of the relations between views. Finally, we conclude in section 4.

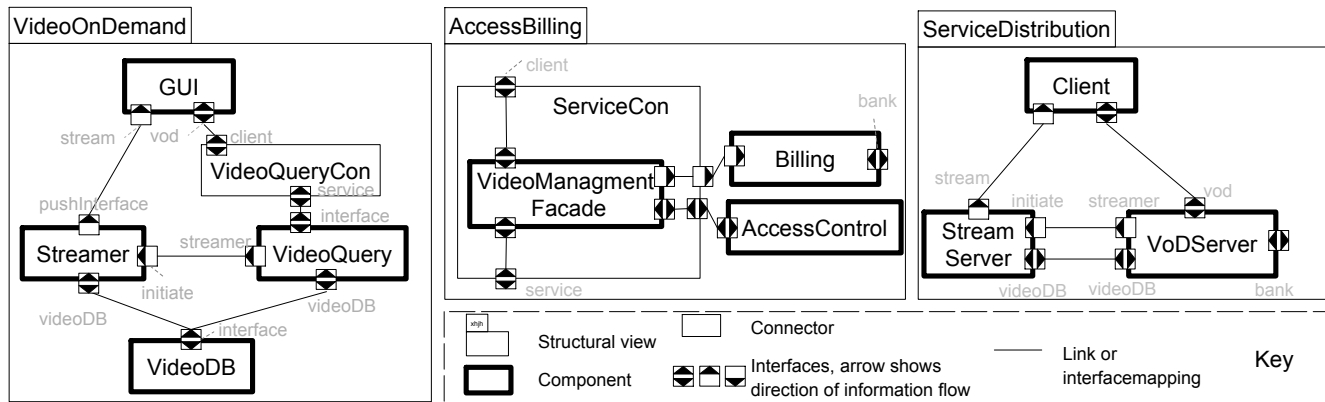


Figure 1. Three basic views for an architecture of a VOD system.

2 Relations and View Composition

We use the architecture of a simplified Video on Demand (VOD) system to illustrate relations and view composition. A VOD system allow users to select and watch video and clip content at their demand. The extract of the architecture we consider includes pay-per-view support, where a user pays for the videos he or she wants to see.

In the extract of the architectural description shown here we focus on three important concerns that influence the structure of the architecture: (1) the basic VOD service that provides support for searching and watching videos; (2) access controlled and pay-per-view, charges are billed to the user's credit card; and (3) the structure of the VOD service in several subsystems to allow easy distribution. The deployment of these subsystems in described in a deployment view (not shown here).

2.1 Producing an architectural description

Figure 1 shows three structural views, each of them covering for one of the concerns identified in the previous paragraph. The VIDEOONDEMAND view structures the system in a GUI component, a Streamer component to handle streams, a VideoQuery component to handle searches and requests and a VideoDB component that manages the database of videos. We assume that each link between two components implies some kind of connector between them, but we only specify the a few important connectors explicitly. The VideoQueryCon connector relays requests to the VideoQuery from the GUI. The ACCESSBILLING view shows components needed for control the access to the server and to bill users for their views. It shows a ServiceCon, linked to a Billing and AccessControl component. Notice that the ACCESSBILLING view can be reused for several applications, the

view does not use concepts of the VOD domain. Finally, the SERVICEDISTRIBUTION view defines the Client component and two components that will contains parts of the VoD service. The architects decided to use a dedicated StreamServer that is optimized for fast data access to the video database and allows fast streaming to clients. The remainder of the VOD system, e.g. searching, managing user accounts, logging on, is encapsulated in the VoDServer component.

Obviously, the three views defined in fig. 1 are related. access control and billing must be applied to the VOD architecture and that the VOD architecture must be allocated to the service distribution. Since intuition alone is insufficient to build solid software, we will explicitly define how the views together form a software system. We illustrate this in section 2.2.

2.2 Relations and view composition

Currently, we have defined three relations, namely unification, refinement and mapping. A unification expresses that elements that appear in the different structural views are the same element. A refinement expresses that a specific structural view describes a substructure for an architectural element of another structural view. Mapping expresses that individual elements or groups of elements (called subjects) from one structural view are subelement of single element of another structural view. A view composition integrates two or more views and relations among these views, resulting in an integrated view. We illustrate the unification and mapping relations and how they are used in composition in the VOD architecture.

To start, we define a composition between the VIDEOONDEMAND and the ACCESSBILLING view, containing a relation that unifies the ServiceCon with the VideoQueryCon. To make the relation specification unambiguous, we specify how the interfaces of

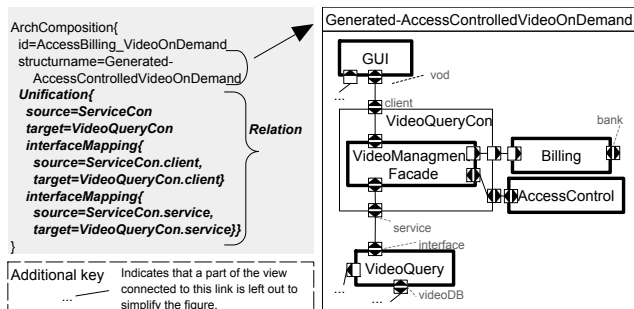


Figure 2. Composition specification and the resulting integrated view.

VideoQueryCon are mapped on ServiceCon with interfaceMappings tags. When a unification is used in a view composition, the element and its interfaces will be named using the names of the target. Next to using relations, a composition specifies the name of the integrated view using the structurename tag. The left hand side of fig. 2 contains the composition specification using simplified syntax, the actual specification has a xml representation. The right hand side shows the integrated view called GENERATED-ACCESSCONTROLLEDVIDEOONDEMAND.

Next, we define a composition between the GENERATED-ACCESSCONTROLLEDVIDEOONDEMAND, and the SERVICEDISTRIBUTION view. This composition exists of several mappings and unifications. The Client and GUI component are unified with each other. The VideoDB and the Streamer are mapped on the StreamServer component, the remainder of the components is mapped on the VoDServer component. The top of fig. 3 shows the composition. Only one of the relations is shown in detail for space reasons, the other relations are specified in a similar way. The bottom of the figure shows the integrated view called GENERATED-DISTRIBUTEDVOD.

2.3 Reflection

We followed two phases during architectural development: (1) defining the basic structures, i.e. the structures in fig. 1 covering the main concerns identified in section 2.1 ; and (2) defining compositions between these structures and generating the integrated structures, i.e. in fig. 2 and 3. The relations between the views are unambiguously defined up to level of interfaces. Architects can make an architectural documentation that includes both the basic views, the important relations and compositions and the resulting integrated views.

More details on the VOD systems and several other examples, including a multi-agent architecture for an Auto-

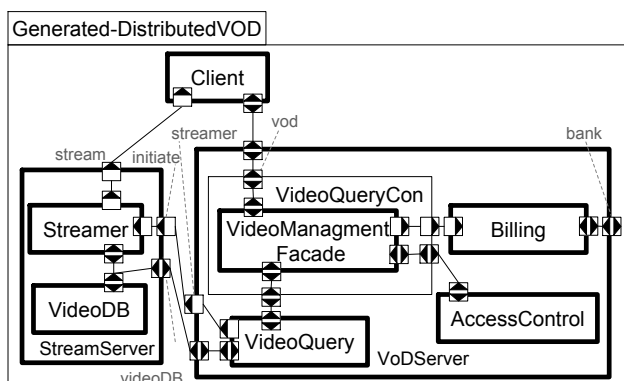
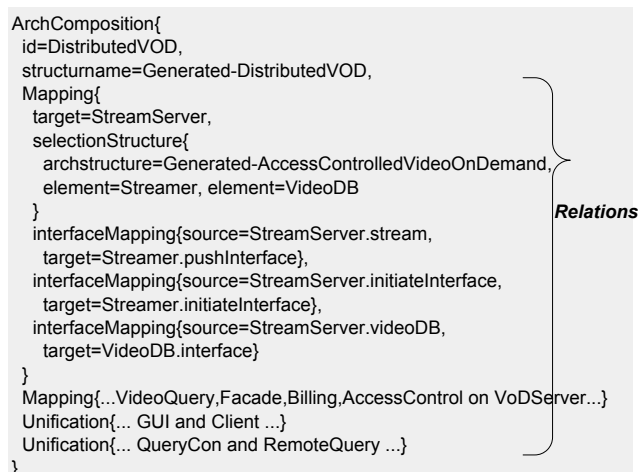


Figure 3. Composition specification and the resulting integrated view.

matic Guided Vehicle Transportation System, can be found on [4].

3 xADLComposition: a tool to support view composition in xADL

To support relations and view composition, we extended the xADL language and toolset (ArchStudio [2]). xADL provides an easy way to extend the language definition and the associated tool support for new concepts, allowing fast prototyping.

Language. The xADL language is specified using several xml schema definition files. We extended this by introducing a new schema definition file, called composition.xsd, defining the concepts of architectural composition and extending architectural structures to allow subelements for both components and connectors. The schema definition files are used by the ArchStudio toolset.

Tool. The ArchStudio toolset has a special-purpose specification editor (ArchEdit), representing the architecture in the

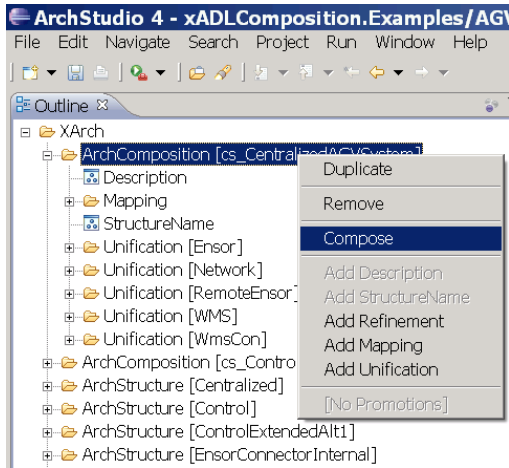


Figure 4. Screenshot of extension to ArchEdit.

form of a tree. The ArchEdit editor can handle any language extensions based on the schema definition files. By defining the compositions.xsd and recompiling the ArchStudio framework with this file, ArchEdit can handle the syntax of relations and compositions. Thus this allows to *specify* relations and compositions in ArchEdit.

To support automated *composition* of structures we alter ArchEdit itself. In our extension, an architect can right-click on a composition specification and ask to automatically generate the integrated view, as shown in figure 4. The tool will then automatically apply the relations and generate an integrated view.

4 Conclusions

We advocate that an architectural description exists of three types of core elements:

- **Views** describe the basic architectural structures of the system.
- **Relations** describe how the views are related to each other. Explicitly defined relations lead to clearer architectural descriptions.
- **Compositions** integrate two or more views and the relations between these views in new integrated views. View composition enables software architects to manage the integration of views and relations between these views in a systematic way. Moreover, view composition can be automated, supporting architects to develop coherent and consistent architectural descriptions.

Our research takes first steps towards realizing this perspective on describing software architectures. As a proof-of-concept, we extended xADL with relations between structural views and view composition, added support for au-

tomatic view composition in the ArchStudio environment, and modeled several example architectures using this extension.

We are working on a formal specification of the semantics of relations and view composition. This specification will enable formal verification of particular properties of a view that must be preserved by architectural composition, such as the provided ports of elements, links between elements, etc. With respect to tool support, view compositions in ArchStudio must be described manually and integrated views must be explicitly generated and are automatically part of the architectural description. For the future, we envision more flexible support, where an architect graphically selects a set of views and associated relations, generate view compositions on the fly and only includes the most useful integrated views in the architectural description.

References

- [1] AADL. <http://www.aadl.info/>.
- [2] Archstudio. <http://www.isr.uci.edu/projects/archstudio/>.
- [3] L. Bass, P. Clements, and R. Kazman. *Software Architectures in Practice (Second Edition)*. Addison-Wesley, 2003.
- [4] N. Boucké. xADLComposition: a tool for view composition in xADL. <http://www.cs.kuleuven.be/~nelis/xADLComposition>.
- [5] N. Boucké, D. Weyns, K. Schelfhout, 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.
- [6] 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.
- [7] E. M. Dashofy, A. van der Hoek, and R. N. Taylor. An infrastructure for the rapid development of xml-based architecture description languages. In *Proceedings of the 24th International Conference on Software Engineering (ICSE2002)*, 2002.
- [8] E. M. Dashofy, A. van der Hoek, and R. N. Taylor. A comprehensive approach for the development of modular software architecture description languages. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 14(2):199–245, 2005.
- [9] D. Garlan, R. T. Monroe, and D. Wile. ACME: Architectural description of component-based systems. In *Foundations of Component-Based Systems*. Cambridge University Press, 2000.
- [10] IEEE. Recommended practice for architectural description of software-intensive systems (ANSI/IEEE-Std-1471), September 2000.
- [11] M. A. Jackson. Some complexities in computer-based systems and their implications for system development. In *Proceedings of CompEuro90*. IEEE Computer Society Press, 1990.
- [12] N. Rozanski and E. Woods. *Software Systems Architecture*. Addison Wesley, 2005.