# Using View-Based Models to Formalize Architecture Description *

Kurt Lichtner, Paulo S.C. Alencar, Donald Cowan
Department of Computer Science
University of Waterloo, Waterloo, Ontario, Canada
{kurt, alencar, dcowan}@csg.uwaterloo.ca

**Abstract**

*Over the last few years, architecture description languages (ADLs) have become both progressively more expressive and widespread. While some ADLs have formally defined semantics, most are still informally specified. In this position paper we present an approach to modeling architecture description languages based on* views. *Our approach is based on partitioning and formalizing each semantically independent concept, or view, represented in an ADL. Along with the approach, we discuss three practical applications of our model, namely ADL comparison, architecture analysis, and component selection.*

**Keywords**: Architectural description languages (ADLs), architectural views, formal models, theory-model paradigm, object-oriented notations.

## 1 Introduction

Within the software architecture community there has been significant emphasis on developing more expressive means of describing architectural designs. Much of the work has been focused on developing new architecture description languages (ADLs), or enhancing the capabilities of existing ones. There has been comparatively less attention aimed at developing a comprehensive formal model of architecture description (i.e., an architectural *meta-model*) that precisely describes the underlying concepts and rules defining the semantics of these languages. This undoubtedly reflects the community's lack of agreement over exactly which categories of information are required in an architectural specification. With the broad range of features and constructs supported by the various ADLs it is a difficult task to find and formalize a *single*, universally appealing model of architectural description. In this paper we outline an approach to formalizing multiple models of description based on the notion of architectural views [1].

* The work described here has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Information Technology Research Centre of Canada (ITRC), and IBM Canada.

We argue that aside from an interesting theoretical exercise, there is considerable practical benefit to this work, in particular, as a rigorous basis for ADL understanding and comparison, architecture analysis, and component selection. Additionally, we are investigating the role of the model as a foundation for reasoning about consistency between views, and its application in the development of ADL-to-ADL translators and interchange languages. In the remainder of this position paper we present our approach to modeling architecture description and briefly sketch its applications.

## 2 Architectural Description Models Based on Views

The construction and use of models as a basis for understanding high-level software descriptions is well established [2, 3]. Our approach focuses on identifying and separately formalizing each of the views captured by an ADL. In the context we present[1], a view is a subset of the semantic categories and features supported by an ADL which can be modeled independently from other categories, i.e., they are entity sets which are, in some sense, "stand alone." For example, while there is nearly unanimous consent that an architectural description outlines the structural aspects of the system, ADLs often allow other properties to be expressed. Commonly cited examples include behavioral specification, dynamic topology, module structure, and inter-process communication model. By abstracting the categories and constructs provided by ADLs into separate views we are better able to deal with the size and complexity of the formalization.

The content and boundary of each view is established by close analysis of the ADL source documentation. The method we follow is based closely on the Theory-Model paradigm [4]. The models are composed primarily of two complementary formalizations, a graphical representation using the Object Modeling Technique (OMT) [5], which captures the categories of information we are representing, as well as any relationships and constraints between them. The full detail of the model is then described using the Z specification language [6].

[1] Although views are increasingly mentioned in the literature, they have at present no well-defined or widely accepted definition. It is our hope that this work will serve to help solidify some of the notions and concepts surrounding this topic.

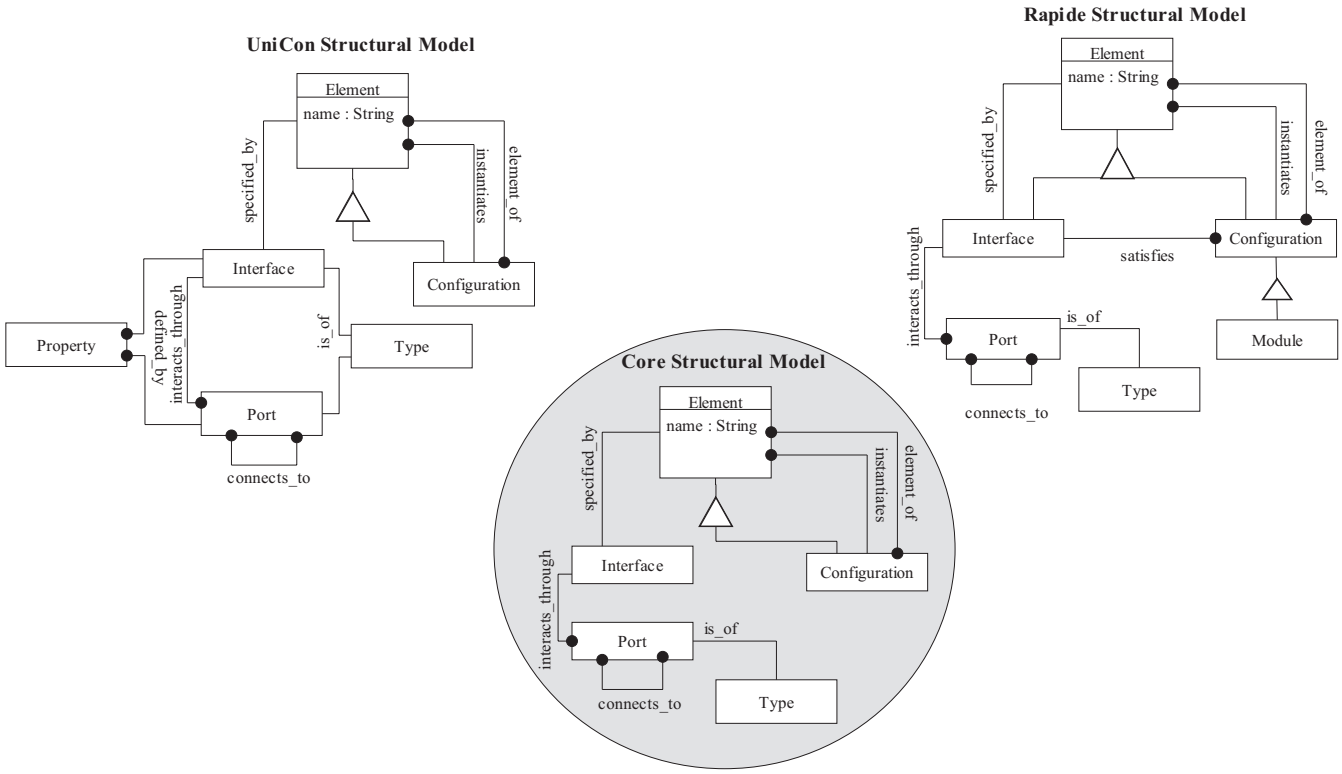


Figure 1: ADL Structural Models

## 2.1 ADL Comparison and Understanding

It is generally accepted that the mainstay of architectural design description is a hierarchically defined configuration of components and connectors [7], or *elements*. This common capability to model structure provides an excellent starting point for formalization. At present, we have completed structural models for Wright [8], UniCon [9], and Rapide [10].

Figure 1 presents the basic structural models[2] for Rapide and UniCon, highlighting their shared core. Individually, these models enable us to explore features of the ADLs which are not readily discernible from their natural language descriptions. From the structural models several core concepts can be identified. As shown in the diagram, we have identified five design categories extant in the basic structural model, *Elements, Configurations, Interfaces, Ports*, and *Types*, along with six relations. Where it enhanced the overall descriptive ability of the model, we combined multiple ADL entities into a single entity within the model. For example, UniCon explicitly supports two symmetric constructs, *Components* and *Connectors*; we found it sufficient to represent both with the category, *Element*. The distinction between the two is more naturally introduced as logical constraints and assertions in the Z description of the model. There are many details specific to the models that have been omitted for simplicity. We do not present the Z formalization in this paper as it is quite detailed.

As the core model is the portion of a view that is invariant across the supporting ADLs, it establishes a foundation for comparing features and constructs of different ADLs. Each language can be characterized by an extension to the core model. This yields a relatively straightforward approach (steps outlined in Figure 2a) for determining precisely how and to what extent each ADL supports the view, and to what extent the ADLs differ in their support for similar features.

[2] Simplified for readability. Full details are provided in [11].

## 2.2 Architecture Analysis

While formal models of programming languages have been used to develop static analysis environments for programs [12], to the best of our knowledge this approach has not been applied to architecture description. The concept, however, is quite similar, consisting of two main components: an architecture knowledge base (AKB) which stores the details of an architectural description, and a facility for querying the knowledge base.

The core models are used to guide both the parsing and analysis of an architectural specification. Elements from within an architectural specification are converted into a data representation based on the formal model. Once the specification has been parsed and the resulting facts stored in a knowledge base, the architect can use the query facility (similar to a standard query language [12]) to ask questions about the design. To illustrate, the following are examples of queries pertaining to the structural aspects of an architectural design (based on the model in Figure 1).

- What is the set of elements in configuration "`conf1`"?

  ```
  Select element.name such that element_of (element, configuration) with configuration = conf1
  ```

- Find all interfaces that interact through more than ten ports:

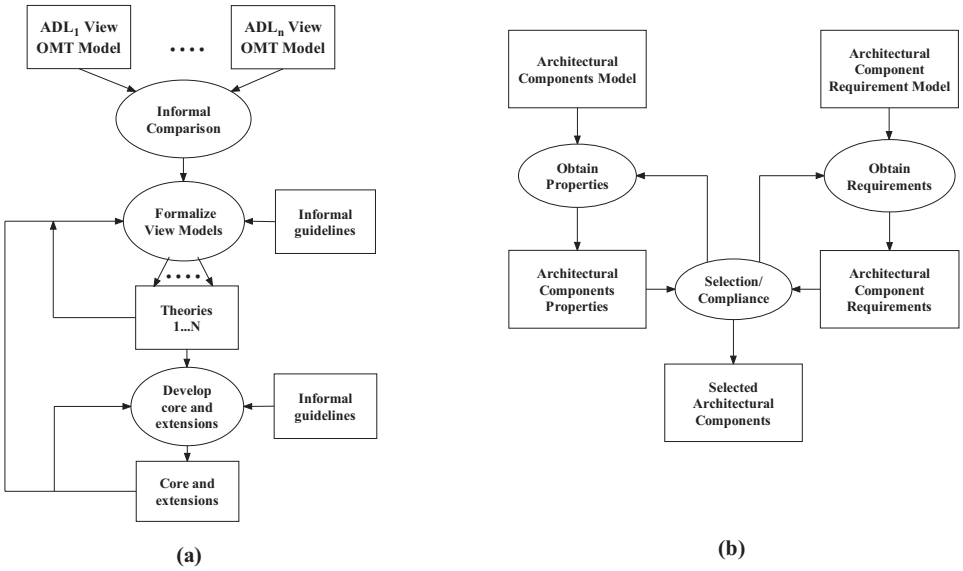


Figure 2: Application overview

`interface i`
`Select i such that interacts_through(i>10,_)`
(where "_" stands for a free entity and the constraint ">10" applies to the number of occurrences of an interface as the first element of a relationship '`interacts_through`').

- Find all interfaces that interact through a port of type "StreamIn":

```
interface i, port p
Select i such that exists [p such that is_of(p,
StreamIn) and interacts_through(i,p)]
```

Since the core models are independent of any particular ADL, this approach has the property that the contents of multiple descriptions of the architecture, specified with different ADLs, can be combined into the same knowledge base.

### 2.3 Architectural Component Selection

Architecture meta-models can also be used as a basis for architectural component evaluation and selection. Our purpose here is to establish the degree of compliance between an architecture component and some specific architectural requirements (an overview is shown in Figure 2b). This particular application was inspired by a selection approach for commercial-off-the-shelf (COTS) software [13]. The elements within our solution include architectural component properties and specific architecture requirements, both of which are described using the model's entity sets and relations. Also required are a set of compliance analysis and checking rules.

Compliance is achieved by validating the properties of the component against the specified architectural requirements. An inference engine such as Prolog, can be used to encode the elements of the model, the specification of the component, as well as the compliance checking rules. Using these rules, the inference engine can validate whether or not any of the required properties are included within the component specification. In other words, it is possible to verify whether a component is applicable for a particular architecture by establishing whether or not it meets a specific set of architectural requirements.

## 3 Looking Ahead

Our immediate goal is to establish models of the various aspects of architectural description by working with the state of the art in ADLs. In addition to the structural view, we are currently constructing models for behavior specification, hierarchical configuration construction, and dynamic topology. Understanding precisely what ADLs express and how they differ requires a formal description of their semantics. In this paper we have presented three main applications which use this foundation.

## References

[1] Paul Clements and Linda Northrop, "Software Architecture: An Executive Overview", Tech. Rep. CMU/SEI-96-TR-003, Software Engineering Institute, Feb. 1996.

[2] M. Rice and S. Seidman, "A Formal Model for Module Interconnection Languages", *IEEE Transactions on Software Engineering*, vol. 20, no. 1, pp. 88–101, Jan. 1994.

[3] Thomas R. Dean and David A. Lamb, "A Theory Model Core for Module Interconnection Languages", in *Proceedings of CASCON'94 – Integrated Solutions*, Toronto, Ontario, Oct. 31–Nov. 3, 1994, IBM Centre for Advanced Studies, pp. 1–8.

[4] Xiaobing Zhang, *A Rigourous Approach to Comparison of Representational Properties of Object-Oriented Analysis and Design Methods*, PhD thesis, Queen's University, Aug. 1997.

[5] M. Rumbaugh, M. Blake, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.

[6] J. M. Spivey, *The Z Notation: A Reference Manual*, ISICS. Prentice-Hall, 2nd edition, 1992.

[7] David Garlan (editor), "Proceedings of the First International Workshop on Architectures for Software Systems", Tech. Rep. Carnegie Mellon University CMU-CS-95-151, Apr. 1995.

[8] Robert Allen and David Garlan, "The Wright Architectural Specification Language", Technical Report CMU-CS-96-TB, School of Computer Science, Carnegie Mellon University, Pittsburgh, Sept. 1996.

[9] Mary Shaw, Robert DeLine, Daniel V. Klein, Theodore L. Ross, David M. Young, and Gregory Zelesnik, "Abstractions for Software Architecture and Tools to Support Them", *IEEE Transactions on Software Engineering*, vol. 21, no. 4, pp. 314–335, Apr. 1995.

[10] David C. Luckham and James Vera, "An Event-Based Architecture Definition Language", *IEEE Transactions on Software Engineering*, vol. 21, no. 9, pp. 717–734, Sept. 1995.

[11] Kurt Lichtner, Paulo Alencar, and Donald Cowan, "Formalizing Architecture Description Languages", Tech. Rep. CS-98-07, University of Waterloo, July 1998.

[12] Stan Jarzabek, "Design of Flexible Static Program Analyzers with PQL", *IEEE Transactions on Software Engineering*, pp. 197–215, Mar. 1998.

[13] Neil A. Maiden and Cornelius Ncube, "Acquiring COTS Software Selection Requirements", *IEEE Software*, vol. 15, no. 2, pp. 46–56, Mar./Apr. 1998.