"Don't Ask, Don't Tell" Inference: Architectural Views and their Interfaces*

Rich Hilliard
ConsentCache Inc.
rh@ConsentCache.com

March 15, 2001

Abstract

This position paper asks the theoreticians for help with a practical class of problems in inconsistency management: integration of multiple architectural views of a system. I outline the problem, suggest a metaphor for approaching the problem and ask some questions.

Introduction

I'm not an expert in non-classical logics. I don't have a general theory to apply to the issues posed by this workshop. Instead, I'm a software architect. I'm interested in practical solutions to inconsistency for a certain class of problems.

In this position paper, I describe that class of problems and some intuitions that may be relevant. I'm writing this position paper in the hope that these issues will generate some interest among the theoretical community; because they are real and perhaps more tractable than the general problem of inconsistency among software artifacts at-large.

Architecural Descriptions

Most modern architecting techniques [1, 3, 6, 7, 8, 9] start from a notion of multiple architectural views as a fundamental, organizing principle of *architectural descriptions* (ADs): those work products used to document the architecture of a software-intensive system.

However, there are virtually no practical techniques for view integration or inconsistency management in this community. My interest is the integration of multiple architectural views into an architectural description for a system of interest. The system may be an individual system, a product line or product family with variants, a system-of-systems, etc.

The architectural views have this character:

- each view is typically written in its own viewpoint language;
- in practice, views are developed somewhat autonomously to solve distinct architectural problems (system structure, security, distribution);
- therefore, the subjects addressed by each view are somewhat orthogonal;

*Submission to the 2nd International Workshop on Living With Inconsistency

- architectural views are “contemporaneous” (or horizontal): issues of vertical refinement do not arise (until later).

Note: I use the terminology of IEEE 1471, which distinguishes views and viewpoints roughly as follows (cf. [10]):

view : viewpoint :: object : type

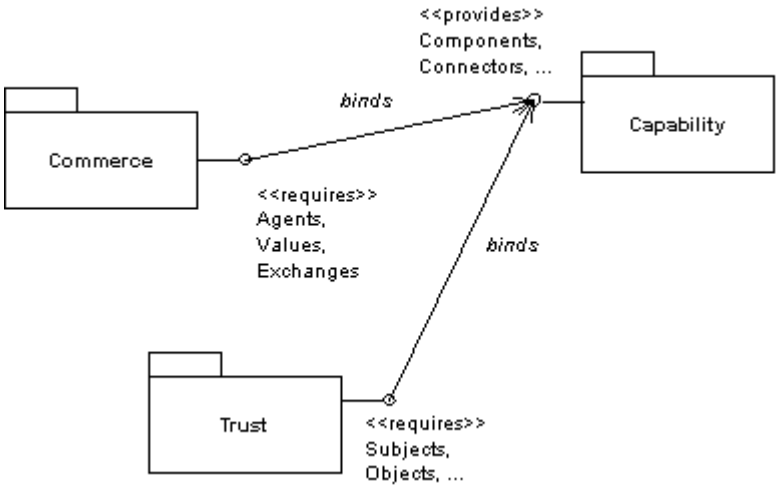
Views as Modules

The character of the views as described above, makes the metaphor of *Views as modules* attractive [4, 5]:

views : architectural description :: modules : program

Modules have these characteristics: (**i**) they facilitate separation of concerns; (**ii**) they promote encapsulation; and (**iii**) module integration can be defined in terms of modules’ interfaces. These notions date back at least to Parnas [11]. The metaphor suggests these ideas: that views, like modules, address separate concerns; that to the extent that views can be non-interfering, those views “encapsulate” concerns; and that if we had some notion of “view interface,” it may be useful to talk about view integration via interfaces.

Pursuing the metaphor further, we might characterise interfaces in terms of the resources they provide and require from other views: a **provides** interface exports viewpoint elements to be used by other views. A **requires** interface identifies “formal parameters” of the view which must be supplied by another view. In the example, the Capability View provides components and connectors. The Commerce View requires that actors and values be instantiated from outside the view, etc.



Questions for the Workshop

Here are some questions the *Views as modules* metaphor suggests:

- What criteria can be used to maximise the independence of views? Of viewpoint languages?
- Are there logics that can underwrite an open-ended set of viewpoint languages to facilitate reasoning about the resulting views?

- Is there a notion of view interface that is useful when reasoning – rather than pure description – is the goal?
- Are interface theories sufficient to write correspondence relations between views [2]?
- Does the **provides**, **requires** paradigm from module interconnection languages suggest a metaphor for view interfaces that can support reasoning?

References

[1] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison Wesley, 1998.

[2] E. Boiten, J. Derrick, H. Bowman, and M. Steen. Constructive consistency checking for partial specification in Z. *Science of Computer Programming*, 35(1):29–75, September 1999.

[3] D. E. Emery, R. Hilliard, and T. B. Rice. Experiences applying a practical architectural method. In A. Strohmeier, editor, *Reliable Software Technologies–Ada-Europe '96*, number 1088 in Lecture Notes in Computer Science. Springer, 1996.

[4] R. Hilliard. Using the UML for architectural description. In R. France and B. Rumpe, editors, *≪UML≫'99 The Unified Modeling Language, Second International Conference*, volume 1723 of *Lecture Notes in Computer Science*, pages 32–48. Springer, 1999.

[5] R. Hilliard. Views as modules. In B. Balzer and H. Obbink, editors, *Proceedings Fourth International Software Architecture Workshop (ISAW-4), 4 and 5 June 2000, Limerick, Ireland*, pages 7–10, 2000.

[6] C. Hofmeister, R. Nord, and D. Soni. *Applied Software Architecture*. Addison-Wesley, 1999.

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

[8] International Organization for Standardization. *ISO/IEC 10746 1–4 Open Distributed Processing—Reference Model (Parts 1–4)*, July 1995. ITU Recommendation X.901–904.

[9] P. B. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 28(11):42–50, November 1995.

[10] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994.

[11] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *CACM*, 15:1053–58, December 1972.