Abstraction, composition, and separation of concern in architecture

Mika Cohen and Ulrik Franke

FOI Swedish Defence Research Agency

Abstraction, composition, and separation of concern are fundamental to architecture: an architecture is a consistent collection of views that deconstruct a system by means of abstraction-realisation, composition-decomposition and separation of concern (cf. [1, 2]). There seems to be considerable confusion, however, within the architecture community as to how these concepts relate to each other. Does composition-decomposition lead to a separation of concern? Many authors claim so (cf. [4, 5]). Does abstraction-realisation similarly lead to a separation of concern? Again, several authors suggest this (cf. [3, 4]). However, these questions have not, as far as we are aware, been explored in a detailed, formal manner in the literature.

In this talk, we try to do so. We answer the questions by means of a rigorous, formal semantics for abstraction, composition, and separation of concern. Using the semantics, we show that composition-decomposition indeed ensures a separation of concern, but not so abstraction-realisation, at least not in general. Thus, confusing composition-decomposition with abstraction-realisation, as is often done, may lead to an unsuccessful separation of concern, as we illustrate with real-world examples from enterprise architecture.

To begin with, we reduce separation of concern to cross-view consistency, another fundamental notion in architecture (cf. [2, 6]). Roughly, given some agreed upon common view, two concerns are separated if any view that addresses the first concern is consistent with any view that addresses the second concern, as long as each view is consistent (by itself) with the agreed upon common view. Hence, the two concerns can be addressed independently, e.g., by two independent projects that each maintain cross-view consistency with the agreed upon common view.

Continuing, we reduce cross-view consistency between an abstract (or composite) agreed upon common view and a realisation (or decomposition) view to syntactic well-formedness constraints (expressible in e.g. OCL) that can be enforced by any conventional architecture tool. Roughly, an abstract view must mirror the realisation view: for every relation between entities in the realisation view, there must exist a corresponding relation between corresponding (more abstract) elements in the abstract view. By contrast, a composite should preserve relations from its decomposition: if a part is related to some entity, then the composite should also be related to that same entity.

Finally, given these definitions, we prove, for a simple process language with input- and output activities, that correct composition-decomposition ensures separation of concern – i.e., any two activities in a process diagram can be decomposed independently – while correct abstraction-realisation does not in general ensure separation of concern – i.e., in general, if two activities are realised independently then the two realisations may be incompatible with each other.

The above result illustrates the importance of distinguishing between abstraction-realisation the one hand and composition-decomposition on the other hand. Perhaps an architect who has seen separation of concern in process modelling achieved with composition-decomposition might mistakenly conclude – if he or she confuses composition-decomposition with abstraction-realisation – that separation of concern in process modelling can also be achieved with abstraction-realisation. Indeed, we show that this mistake has been made in enterprise architecture projects in industry.

References

1. Dewayne E. Perry and Alexander L. Wolf: Foundations for the Study of Software Architecture. SIGSOFT Software Engineering Notes, 17(4):4052, 1992.
2. ISO/IEC/IEEE 42010:2011, Systems and software engineering – Architecture description
3. Marc M. Lankhorst, René van Buuren, Diederik van Leeuwen, Henk Jonkers, Hugo ter Doest: Enterprise Architecture Modelling – the Issue of Integration. Adv. Eng. Inform. 18(4):205216, 2004.
4. Alexander Pretschner, Manfred Broy, Ingolf H. Kruger, and Thomas Stauner. Software Engineering for Automotive Systems: A Roadmap. Future of Software Engineering (FOSE '07). 2007.
5. Stephan Kurpjuweit and Robert Winter: Viewpoint-based Meta Model Engineering. Proceedings of the 2nd Int'l Workshop EMISA. 2007.
6. Ivano Malavolta, Patricia Lago, Henry Muccini, Patrizio Pelliccione, and Antony Tang: What Industry Needs from Architectural Languages: A Survey. IEEE Trans. Softw. Eng. 2013.