Dealing with concerns ask for an architecture-centric approach

Nelis Boucké and Tom Holvoet
AgentWise, Distrinet, Departement of Computer Sience, KULeuven, Belgium
{nelis.boucke,tom.holvoet}@cs.kuleuven.be

It is becoming more and more clear that architectural design is a crucial step in building complex software. Architectural design involves the first and most crucial design decisions to meet the expected quality attributes [1]. Dealing with concerns that influence important quality attributes asks for an architecture-centric approach. In this paper we propose architectural concerns as basic concept for such architecture-centric approach and discuss possible ways of representing architectural concerns.

Software architecture and concerns

The requirements and the domain model describe the problem, software architecture is the solution to the problem. The software architecture defines the general structure of the system (the solution) to meet functional requirements and satisfying essential quality requirements (the problem).

Concerns in software systems are in essence 'issues important for the stakeholders of the software system'. The software architect, being an important stakeholder and responsible for building the software architecture, will have his own set of essential issues. From this *architectural concerns* are defined as 'the concerns of the software architect'. Example architectural concerns could be coordination between distributed entities or security guarantees of the system.

By narrowing down the definition to 'concerns of the architect', there is a distinction between requirements and architectural concerns: requirements describe the problem, architectural concerns are issues of the solution. There is no perfect alignment between requirements and architectural concern, because the concerns of a software architect are not only influenced by the requirements but also by the solution strategy. Thus a single architectural concern will contribute (positively or negatively) to the fulfillment of several requirements.

Representation of architectural concerns

Until now only the definition of architectural concerns has been discussed. Another important issue is how architectural concerns can be represented to prevent scattering and tangling [8] of the representation of an individual concern (and thus allow good separation of concerns). Architectural concerns can be represented in several ways, depending on the complexity of the concern (e.g. amount of software entities, relation complexity) and the goal of the architect. Moreover, the representations do not fully exclude each other. Some possible ways of representing architectural concerns are:

1. Component aspects
2. Instantiated architectural patterns
3. Architectural views

1. Component aspects are a logical extension of code level aspects [8, 11, 2]. Component aspects are well suited to modularize solutions in a *single aspect component* with similar characteristics as components and within a *single viewpoint* on the system. Examples from literature are [12, 13, 10].

2. Architectural patterns are solutions with well known properties, involving *several software entities and the relations* between them. Instantiating patterns permits for identifying solutions in isolation and composing those solutions to build the final software. Architectural patterns are typically used within a *single view*. Patterns are useful when well known solutions exist. Care should be taken in combining the patterns to prevent interference between them, both for functional as for quality attributes [5].

3. An architectural description uses several architectural views [9, 4]. Such a view is a partial description of the software architecture from a certain perspecive, to emphasizing certain facets of the solution as clear and as concise as possible while deemphasizing and ignoring other facets. Each view uses some style, providing the set of abstractions that can be used in the view (software elements, externally visible properties and relationships between them [1, 5]). Compared with the previous approaches, views inherently involve *multiple viewpoints* on software architecture and typically involve *many software entities and relations* to each other.

Architectural views are our main interest for two reasons. (1) Using views to represent concerns is close to current practice in software architecture, views have been introduced to represent different stakeholders concerns and different types of structures. Architectural views could be seen as an MDSOC [14] mechanism on architectural level [7, 6]. But the current notion of software architectures lacks a structured way to describe non-trivial relationships between views. Some implicit, informal or simple relationships (e.g. one-to-many of software elements) may be feasible for typical styles like component- connector and module-decomposition, there relation is common knowledge in software engineering. But simple relations are insufficient for new types of views (containing an architectural concern) having non-trivial relationships and probably intefering with other views. We believe that new mechanisms are needed to describe the composition of architectural views (for architectural views) and to translate those views to detailed design. In this context, we provided an initial sketch for State-based join-points [3] to describe the relation between the 'coordination' architectural concern and the remainder of the application.

(2) We believe architectural concerns show interesting properties to represent more 'coarse grained' architectural concerns. There could arise some confusion because architectural views do not formally 'modularize' concerns, architectural views are not 'modules' of the final software system! But notice that the description of different architectural concerns is fully separated (thus tangling and scattering of representations is prevented) and the different architectural concerns can be recomposed to make up the final software architecture, clearly a mechanism of separation of concerns.

Kandé et al. [7, 6] describes one possible way of using architectural views to describe concerns, called Perspectival Concern-Space (PCS). PCS is an interpretation of the IEEE-Std-1471 and uses an extension of UML to describe concerns.

Many open questions regarding the nature of architectural views to represent architectural concerns remain open, e.g.:

1. Which architectural concerns should be described as views?
2. How do new architectural views relate to the classical views like module decomposition, component and connector, ...?
3. What concepts of AOSD apply on architectural level? Weaving architectural concerns? Architectural join-points? Aspects? Or should join-points being described in a radical new way?
4. How do architectural concerns relate AOSD techniques of design and implementation?

References

[1] Len Bass, Paul Clements, and Rick Kazman. *Software Architectures in Practice (Second Edition)*. Addison-Wesley, 2003.

[2] Lodewijk Bergmans and Mehmet Aksits. Composing crosscutting concerns using composition filters. *Communications of the ACM*, 44(10):51–57, 2001.

[3] N. Boucke and T. Holvoet. State-based join-points: Motivation and requirements. In Robert E. Filman, Michael Haupt, and Robert Hirschfeld, editors, *Proceedings of the Second Dynamic Aspects Workshop (DAW05 at AOSD05)*, pages 1–4, 2005.

[4] Paul Clements, Feliz Bachman, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. *Documenting Software Architectures, Views and Beyond*. Addison Wesley, 2003.

[5] David Garlan and Mary Shaw. An introduction to software architecture. In V. Ambriola and G. Tortora, editors, *Advances in Software Engineering and Knowledge Engineering*, pages 1–39, Singapore, 1993. World Scientific Publishing Company.

[6] Mohamed Kandé. *A Concern-Oriented Approach to Software Architecture*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2003.

[7] Mohamed Mancona Kande and Alfred Stroheier. On the role of multi-dimensional separation of concerns in software architecture. In *Prooceedings of the OOPSLA'2000 Workshop on Advanced Separation of Concerns*, 2000.

[8] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.M. Loingtier, and J. Irwin. Aspect-oriented programming. *Proceedings European Conference on Object-Oriented Programming, Springer-Verslag*, 1241:220–242, 1997.

[9] P. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50, November 1995.

[10] M. Mezini and K. Ostermann. Conquering aspects with caesar. In M. Aksit, editor, *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development (AOSD)*, pages pp. 90–100. ACM Press, 2003.

[11] H. Ossher and P. Tarr. Multi-dimensional separation of concerns and the hyperspace approach. In *Proceedings of the Symposium on Software Architectures and Component Technology: The State of the Art in Software Development*. Kluwer, 2000.

[12] Monica Pinto, Lidia Fuentes, and Jose Maria Troya. Daop-adl: an architecture description language for dynamic component and aspect-based development. In *GPCE '03: Proceedings of the second international conference on Generative programming and component engineering*, pages 118–137. Springer-Verlag New York, Inc., 2003.

[13] D. Suve, W. Vanderperren, and V. Jonckers. Jasco: an aspect-oriented approach tailored for component based software development. In *In Proc of international conference on aspect-oriented software development (AOSD)*, 2003.

[14] Peri L. Tarr, Harold Ossher, William H. Harrison, and Stanley M. Sutton Jr. N degrees of separation: Multi-dimensional separation of concerns. In *International Conference on Software Engineering*, pages 107–119, 1999.