On the use of Specification Styles in the Design of Distributed Systems

Chris A. Vissers, Giuseppe Scollo, Marten van Sinderen, Ed Brinksma

University of Twente, Fac. Informatics
7500 AE Enschede, NL

Abstract

The paper uses the term architecture to denote an abstract object that is defined in terms of a set of requirements for a product and that is used to derive various concrete objects, called implementations, from it. It is assumed that an architecture is expressed in a formal description language. The paper argues that in practice any architecture of more than elementary complexity, and thus its formal description, needs to be structured to keep it comprehensible and to efficiently express its functionality. This structuring introduces implementation oriented elements in the architecture, despite the fact that in principle the architecture should be implementation independent: i.e. it should be just a definition of the abstract object's external functionality.

The necessity of structuring formal descriptions implies a de facto responsibility of the architect for the equality of the implementation. To exploit this responsibility the architect should should obey qualitative design principles like ortogonality, generality, generality and open-endedness. Substanctial experience with the development of distributed system architecture has shown, however, that the task of finding an appropriate structure is seriously underestimated in terms of complexity and time required.

The paper, therefore, advocates the use of well defined specification styles that allow to structure formal specifications and can be used at advantage to pursue qualitative design principles. The establishment of common and related specification styles is also considered paramount to preserve homogeneity of large specifications developed by teams of specifiers. Such specification styles would enable the designer to control better the design trayectory and thus to produce higher quality designs in shorter timescales.

The paper introduces a monolithyc, a constraint oriented, a state oriented, and a resource oriented specification style. For each style it is globally indicated which design objectives may be supported. The styles can be used to support the complete design trayectory from architecture to implementation, as is also formally demonstrated by a verification exercise in the context of a simple example. The formal background of the constraint oriented style is explained in more detail. The style defines an object in terms of a set of constraints, each of which can be chosen so as to correspond closely to a natural element of human reasoning about the behaviour of an object. The constraint oriented style, therefore appears to be very effective in initial design phases when requirements capturing is the dominant objective.

The reasoning in the paper is using the specification language LOTOS to convey ideas and to present examples, but can be applied to other languages that support this styles.