Designing Software Architectures as Knowledge Specializations

Invited Paper for the Panel Role of Architecture in the Development of Software Systems

Mehmet Aksit
University of Twente, Department of Computer Science
P.O. Box. 217, 7500 AE Enschede, The Netherlands
aksit@cs.utwente.nl, wwwtrese.cs.utwente.nl/~aksit

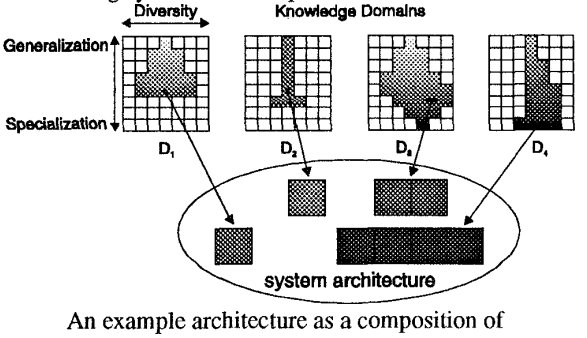
1. Object-Oriented Software Development

Object-oriented methods aim at providing natural ways for decomposing (or composing) a system into (from) objects that correspond to concepts in the customer's problem domain. The identified objects are the basic building blocks of the object-oriented system to be constructed. In order not to disregard relevant objects, most methods advise software engineers to take dedicated steps such as reading books about the problem domain, interviewing customers, etc.

We consider two important concerns in understanding the problem domain. First, it is very important to identify all the objects that are required for defining a consistent system, at least in its minimum configuration. Second, identified objects must serve as composable building blocks to construct robust, adaptable and reusable *architectures*. It has been shown in [2] that the choice of architectural style can have far-reaching consequences because they shape the analysis of the problem and the expression of the design.

2. Designing Object-Oriented Architectures

The following figure illustrates our architecture concept. This architecture consists of 4 components which are considered necessary in providing the expected behavior of the software system. As an example, these components may represent the engine, chassis, breaking and steering systems of a simple motorized vehicle.



An example architecture as a composition of specialization of knowledge domains D1 to D4.

The architecture is a particular composition of specializations from the related knowledge domains D_1 to D_4. Each knowledge domain is modeled as a matrix. Rows and columns represent generalization-specialization and diversity relations among matrix elements, respectively. Each element in a matrix represents a concept in the corresponding knowledge domain. Each row –except the top row– is a specialization with respect to its higher adjacent row.

3. Pilot Projects

To verify our architecture concept, we carried out 3 pilot projects, namely on transaction processing, image processing and fuzzy-logic based reasoning. We implemented and tested these architectures extensively. At this stage we experienced three problems. First, we spent a considerable amount of time in searching and understanding the related knowledge domains. Nevertheless, in all the pilot projects, we could extract satisfactory information from the literature. Second, sometimes it was necessary to extend domain knowledge to make it suitable for architecture definition. Third, we realized that mapping knowledge into object-oriented concepts was sometimes difficult, because certain aspects of knowledge could not be represented directly in terms of object-oriented concepts. Our conclusion is that the architecture concept is the right choice for developing robust, adaptable and reusable software systems. Mapping problems to object-oriented models can be solved by using more advanced object-oriented models than the ones provided by standard methods or languages. More detailed information about our approach is given in [1].

References

[1] M. Aksit et al. Designing Software Architectures as a Composition of Specializations of Knowledge Domains. University of Twente, Memorandum no: 95-44, 1995.

[2] M.Shaw, Making Choices: A Comparison of Styles for Software Architecture, IEEE Software, Vol 12, No 6, November 1995, pp. 27-41.

0730-3157/96 $5.00 © 1996 IEEE