Putting Formal Description of Software Architecture in Practice: Good News, Bad News.

Paola Inverardi

Dipartimento di Matematica Pura ed Applicata, AREA INFORMATICA
Universitá dell'Aquila , Via Vetoio, 67100 L'Aquila, Italy
Tel. +39 0862 433127, Fax +39 0862 433180,inverard@univaq.it

Since a few years the research group in L'Aquila has been carrying on work in the field of architectural design. Our first activity has been to face the problem of formally defining an SA description. In this context the use of a rewriting-based specification language, the Chemical Abstract Machine, already known in the literature, to describe software architectures has been proposed [7, 8]. There were several reasons to attack the problem of suitably describing Software Architectures with a formal approach. Software Architectures exhibit both static and dynamic features, have to be described at a high level of abstraction, and must be comprehensible to a variety of *users* who have different educational background and different goals. All these requirements suggest Software Architecture descriptions must be precise, non ambiguous, general, abstract and expressive. On the other hand, since SA description have to be as abstract and concise as possible, the SA field seemed to be a good benchmark on which formal techniques might show their usefulness as opposed to the specification field, in which the scaling up problem blocked, de facto, the use of formal techniques in real world projects. Our approach has been shown to be quite effective in terms of verification and analysis of software architecture properties, both behavioral [7, 8] and quantitative [5, 6, 3]. Recently, the architectural description have been used to derive or to retrieve information useful to devise integration testing plans [1, 2]. In all these cases we could rely on a formal SA description that we developed, in one of the above cases we dealt with a real case study, that is the SA description of a system we specified was then used to design the actual system implementation in a standard refinement development strategy. As a matter of fact all the system descriptions we worked on were quite small and could be easily formally analyzed both algebraically and through model checking techniques.

Since a couple of years we got involved in joint projects with three different telecommunications companies, in all the projects we were asked to work on the SA description of (part of) one of their product system. We started then experiencing reverse engineering problems. All our customers wanted as ultimate goal to achieve a clear and effective SA description in order to conduct predictive analysis and evaluation of the architectural choices. In most cases their primary interest was in predicting system performance behavior. One of them was also interested in evaluating and measure changes impact on the architectural design. Another was also interested in obtaining a complete and clear documentation of the system.

The first problem we had to face in these cases was the choice of the ADL. We never thought of using formal ADLs, either Cham or others, mainly for a problem of standard maturity. We decided to use UML [15, 16] as our driven technology. There are several motivations for this choice. The first one is that we had to use a standard and tool supported set of notations to carry out the documentation phase. The second is that several authors used it in architectural descriptions even for large systems [12, 13, 14]. Last but not least, for the environments we were referring to the use of UML was already a big deviation step from their standard developing process.

All the systems we dealt with are event–based telecommunication systems, written in proprietary languages. The peculiarity of this kind of systems is that the most complex part of their model understanding deals with the dynamics as opposed to the static system structure. From a static structural point of view the source code is rather simple in terms both of internal control flow and of data structures complexity. On the contrary the components interactions can be extremely complex and deeply intertwined with operating systems and hardware communication protocols.

UML has been used to model the static description (using stereotyped Class diagrams) and components interactions (using Sequence and State Diagrams). Obviously this showed to be quite unsatisfactory with respect to the dynamic modeling. In one case, we tried to introduce besides UML, formal descriptions as well. We were in fact interested in predictive performance analysis and we were trying to apply our approach [6, 3] for the derivation of performance model. This approach requires the construction of the global system finite state model. We used a Labelled Transition System (LTS), based on the FSP [10, 9] specification, generated using the LTSA tool [11]. This modeling technique although useless to analyze the system global behaviour (due to state explosion) resulted in an interesting tool to analyze subsystem behaviour. To overcome the state explosion problem we are defining a way to get the information needed for the performance model out of a set of sequence diagrams enriched by state information retrieved from the single component state diagram [4].

Summarizing on our experience, we are interested in discussing the role of formal SA descriptions in real projects as it can be realistically proposed nowadays. We believe that standard techniques should be used to the maximum extent and that formal techniques should be coherently combined with them.

References

[1] A. Bertolino, P. Inverardi, H. Muccini, A. Rosetti. An Approach to Integration Testing Based on Architectural Descriptions. *IEEE Proc. ICECCS-97*, Como 1997.

[2] A. Bertolino, F. Corradini, P. Inverardi, H. Muccini. Deriving Test Plans from Architectural Descriptions. To appear on *Proc.* 22^{nd} *Int. Conf. on Softw. Eng. (ICSE2000)*, Limerick (Ireland), June 4^{th}-11^{th}, 2000.

[3] F. Aquilani, S. Balsamo, P. Inverardi. An Approach to Performance Evaluation of Software Architectures Internal Report Universita' dell'Aquila, Marzo 2000, submitted for pubblication

[4] F. Andolfi, F. Aquilani, S. Balsamo, P. Inverardi. Deriving QNM from MSCS for Performance Evaluation of Software Architectures Internal Report Universita' dell'Aquila, Aprile 2000, submitted for pubblication

[5] S. Balsamo, P. Inverardi, C. Mangano, F. Russo. Performance Evaluation of a Software Architecture: A Case Study. *IEEE Proc. IWSSD-9*, April 1998, Ise-Shima, Japan.

[6] S. Balsamo, P. Inverardi, C. Mangano, Performance Evaluation of Software Architectures. *ACM Proc. WOSP*, Santa Fe, New Mexico 1998.

[7] P. Inverardi and A.L. Wolf. Formal Specifications and Analysis of Software Architectures Using the Chemical Abstract Machine Model. *IEEE Transactions on Software Engineering*, 21(4):100–114, April 1995.

[8] D. Compare, P. Inverardi and A. L. Wolf. Uncovering Architectural Mismatch in Component Behavior. *Science of Computer Programming* (33)2 (1999) pp. 101-131.

[9] Magee,J. and Kramer, J. Concurrency: State Models and Java programs, Wiley Pub. 1999.

[10] Finite State Processes (FSP) on line at: "http: www-dse.doc.ic.ac.uk/jnm/book/ltsa/Appendix-A.html"

[11] Labelled Transition Systems Analyzer (LTSA) on line at: "http: www-dse.doc.ic.ac.uk/jnm/book/ltsa/LTSA.html"

[12] Jason E. Robbins, Nenad Medvidovic, David F. Redmiles and David S. Rosenblum. Integrating Architecture Description Languages with a Standard Design Method. *Proc. 20th Int'l Conf. on Software Engineering* Apr. 1998, pp. 209-218.

[13] C. Hofmeister, R.L. Nord and D. Soni. Describing Software Architecture with UML. *1st Working IFIP Conference on Software Architecture (WICSA1)*, pp. 145–159, San Antonio, Texas Berlin, Germany, 22-24 February 1999.

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

[15] Rational Corporation. Uml Resource Center. UML documentation, version 1.3. Available from "http://www.rational.com/uml/index.jtmpl".

[16] I. Jacobson, G. Booch and Rambaugh. The Unified Software Development Process. Addison Wesley publiser.