THALES Project No. 1194

FL-ARCH DESIGN: Formal Description Languages for the Architectural Design of Software Systems

Research Team

Manolis Skordalakis, *Professor* [*]

Nikolaos S. Papaspyrou, *Lecturer*

Paris Avgeriou, *Doctoral Candidate* [†]

Andreas Papasalouros, *Doctoral Candidate* [‡]

Kyriakos Ginis, *Doctoral Candidate*

National Technical University of Athens, School of Electrical and Computer Engineering
Software Engineering Laboratory, Polytechnioupoli Zografou, 15780 Athens, GREECE.
E-mail: {skordala, nickie, pavger, andpapas, kyrginis}@softlab.ntua.gr

Introduction

The goal of the research project that is summarized in this paper was the study of *formal software architecture description languages*, concerning their syntax and semantics. The main final outcome was the design of such a prototype language, based on existing, accepted and well documented languages of this research field, as well as the investigation of how to implement architectural design tools that support this language.

Software Architecture intends to improve the software development process and therefore is of paramount importance in the information society. For the description of software architecture, it is necessary to define a suitable formal notation and appropriate tools to support it. *Architecture Design Languages* (ADLs) have been proposed as such a formal notation. This research project contributes to the theoretical study of ADLs and to the solution of related problems that are currently considered as open problems.

Software Architecture

The perception of the field of software architecture varies widely. On the one hand, there are people who consider it as simple 'box and arrow diagrams'. On the other there are those who claim that software architecture is a panacea that will revolutionize software development. Nevertheless industry and academia have reached consensus that investing on architectural design in the early phases of the lifecycle is of paramount

[*] Prof. Manolis Skordalakis retired in September 2003. He is currently a Research Associate of the Software Engineering Laboratory.

[†] Paris Avgeriou completed his doctoral dissertation in January 2003. He is currently a Senior Researcher at the IPSI CONCERT division of the Fraunhofer Institute.

[‡] Andreas Papasalouros completed his doctoral dissertation in May 2004. He is currently a research Associate of the Software Engineering Laboratory.

importance to the project success. Moreover there is an undoubted tendency to create an engineering discipline on the field of software architecture if we consider the published textbooks, international conferences devoted to it, and recognition of architecting software systems as a professional practice.

Despite the attention drawn to this emerging discipline, there has been little guidance, regarding how to describe and document software architecture. Software architecture is nowadays at a similar level of maturity as the architecture of buildings and structures was in the middle of the nineteenth century. Today, as back then, professionals with very different rôles, knowledge and dexterities call themselves software architects regardless of whether their education was that of a software engineer or a programmer's. However, it is rather impossible for software "craftsmen" who are simply adept in design and programming to be able to design the enormous, complex software systems that are typical of our days. The young field of software architecture creates new requirements in software development.

There have been numerous approaches from academia, industry and international bodies, on what the description and documentation of software architecture entails and what process should be followed to perform it. IEEE has developed a Recommended Practice for the architectural description of software-intensive systems, which contains a framework of concepts in order to facilitate the adoption of architectural principles and practices in the industrial and research community. It remains at a general level of prescription but provides a common denominator for tackling the task of architectural documentation. For the specific category of Open Distributed Processing Systems, an ISO/IEC committee has developed the Reference Model for Open Distributed Processing (RM-ODP). This standard guides development teams into designing architectures that support distribution, interoperability and portability. The Open Group has also developed The Open Group Architectural Framework (TOGAF), which supports some of the IEEE 1471 standard concepts and practices.

Documentation of software architecture has always been concerned with the definition of the appropriate notations or languages for designing the various architectural artifacts. As a consequence, a different genre of languages has emerged over the past ten years: Architecture Description Languages (ADLs), which aim at formally representing software architectures [1]. Unfortunately these languages have never been broadly used in industry and most of them lack support by appropriate tools. However the recent trend is the use of the widely accepted Unified Modeling Language as an ADL, either by extending it per se, or by mapping existing ADLs onto it.

Software Architecture Description Languages

Formal notations for describing software architecture and appropriate tools are necessary to support the model of architecture-centric software development. As a formal approach to representing architecture, software architecture description languages present several advantages with respect to non-formal approaches. A formal description of software architecture: (i) is more likely to be closely implemented and maintained by a development team, (ii) is considered as stronger and more official by the development team, (iii) facilitates the communication between the members of the software development team, providing a solid common background, (iv) provides a portable abstraction of the software system that is useful on its own, (v) can be reused in the

development of future software projects, and (vi) provides the basis to automate software design with appropriate supporting tools. For all these reasons, they have evolved as an important research field in Software Engineering and several attempts have been made to classify and evaluate them [2, 3].

The starting point for software architecture description languages was Carnegie Mellon's Software Engineering Institute. Other active research groups have been and/or are located at the Stanford University, University of Texas at Austin, Northeastern University, Ohio State University, University of Southern California, University of California at Irvine, Honeywell Technology Center, Naval Postgraduate School, Teknowledge Ltd. A recent and thorough survey paper also attempted to define the notion of ADLs [4]. Other attempts have been made to define the characteristics and requirements from ADLs [5]. A small list of several ADLs that have been proposed contains: Aesop (Garlan *et al.*), ArTek (Terry *et al.*), SEL (Medvidovic *et al.*), Darwin (Magee *et al.*), LILEANNA (Tracz), MetaH (Binns *et al.*), Rapide (Luckham *et al.*), SADL (Moriconi *et al.*), UniCon (Shaw *et al.*), Weaves (Gorlick *et al.*), and Wright (Allen *et al.*). Also, several of the designers of ADLs have collaborated, aiming to create an architecture interchange language for the mapping of architectural descriptions from one ADL to another [6].

Known problems regarding today's ADLs are: (i) their shortage in describing the interaction between software subsystems, (ii) their poor abstraction capabilities, (iii) their inability to properly describe interfaces, (iv) their bad support for the component software paradigm, and (v) their lacking support in incorporating several programming languages and paradigms, as well as existing legacy systems in the design of new software systems. A current tendency in the field, which has also been used in this research project, is to use the internationally accepted Unified Modelling Language (UML) either per se as an ADL, or by mapping existing ADLs to it [7, 8].

A comparative study of ADLs was performed in this project, in order to identify the characteristics of ADLs and the requirements that are imposed. Figure 1 summarizes the outcome of this task.

- **Components**
  - Interface
  - Types
  - Semantics
  - Constraints
  - Evolution
  - Non-functional properties
- **Connectors**
  - Interface
  - Types
  - Semantics
  - Constraints
  - Evolution
  - Non-functional properties
- **Architectural Configurations**
  - Understandability
  - Compositionality
  - Refinement and traceability
  - Heterogeneity
  - Scalability
  - Evolution
  - Dynamism
  - Constraints
  - Non-functional properties
- **Architectural Analysis**
  - System quality attributes discernable at runtime
  - System quality attributes non-discernable at runtime

Figure 1. Architecture description languages: characteristics and requirements.

A Prototype Architecture Description Language

Throughout this project, a prototype ADL was defined. It was named SEL and was based on the Unified Modelling Language (UML). UML is a general purpose modelling language, supporting the full software life cycle, from requirements to final code. Moreover, UML can provide multiple views of the system and is fully extensible, with a semi-formal semantics, and is supported by a variety of tools. However, UML does not directly provide all the elements that are necessary for architectural modelling, e.g. connectors, architectural configurations and styles.

In this project, we followed the approach of restricting UML's meta-model, in order to focus on the task of architectural design. This was achieved using some of the language's extension mechanisms: constraints, tagged values, stereotypes and profiles. The main advantage of our approach is that it explicitly represents architectural concepts by creating new modelling elements. In this way, software architecture can be described using the tools that support UML and can be understood by UML users. The disadvantage of this approach is the difficulty in defining architectural concepts.

The language SEL is based on C2 [9], but differs significantly in the exchanged messages and the semantics of connectors. The purpose of the language is to provide a linguistic formalism that treats components and connectors as first-class elements. SEL is defined as a UML profile. Constraints are defined using the Object Constraint Language (OCL), which is closely related to UML.

Tools that support software architecture

Although the suitability of an ADL does not depend on the tools that support this language, such tools unquestionably make an ADL more useful in practice. For this reason, many researchers have turned to the study of support tools, in an effort to define their characteristics and to come up with prototype "toolkits". There is a big gap between the properties that are desired from such support tools and what one sees in the software market today. Although several ADLs are provided together with one tool of some sort (C2 and Rapide are two exceptions), these tools focus on individual characteristics, e.g. analysis, refinement or dynamism.

In this project, the characteristics of an ideal support tool for architectural design were studied. A possible implementation of such a tool to support SEL was also investigated, as an extension to the popular development environment IBM Rational XDE.

Conclusions

The primary results of this project can be summarized as follows:

- A thorough study of the scientific literature in the field of software architecture description languages was conducted. The young members of the research team were introduced to this literature.
- A detailed catalogue of formal and informal architecture description languages that have been proposed and/or are used in practice was assembled.
- The basic characteristics and requirements of ADLs were identified and studied.

- SEL, a prototype ADL was designed, based on the identified characteristics and satisfying the identified requirements.
- The implementation of tools to support architectural design with the ADL SEL was investigated.

Most of these results are part of the doctoral dissertation submitted by the primary young researcher who participated in the project. The dissertation was completed after the project's end [D1]. Some results are related to the doctoral dissertation submitted by the second young researcher, also completed after the end of the project [D2]. All the results of the project were disseminated to the scientific community with publications in academic journals [P1, P2, P3] and presentations in international conferences [P4].

Publications

P1. P. Avgeriou, A. Papasalouros, S. Retalis, M. Skordalakis, "Towards a Pattern Language for Learning Management Systems", *IEEE Educational Technology & Society*, vol. 6, no. 2, pp. 11-24, 2003.

P2. P. Avgeriou, S. Retalis, N. Papaspyrou, "Modeling a Learning Technology System as a Business System", *Software and Systems Modeling*, vol. 2, no. 2, pp 120-133, Springer-Verlag, July 2003.

P3. P. Avgeriou, "Describing, Instantiating and Evaluating a Reference Architecture: A Case Study", *Enterprise Architect Journal*, Fawcette Technical Publications, June 2003. Available online from: http://www.ftponline.com/ea/

P4. P. Avgeriou, N. Guelfi, R. Razavi, "Patterns for Documenting Software Architectures", in *Proceedings of the 9th European Conference on Pattern Languages of Programs* (EuroPLOP), Irsee, Germany, July 2004.

Doctoral Dissertations

D1. P. Avgeriou, *A Reference Architecture for Open Learning Management Systems*, Doctoral Dissertation, School of Electrical and Computer Engineering, National Technical University of Athens, January 2003.

D2. A. Papasalouros, *Design Models and Automatic Composition of Courseware*, Doctoral Dissertation, School of Electrical and Computer Engineering, National Technical University of Athens, May 2004.

References

1. P. C. Clements, R. Kazman and M. Clein, *Evaluating Software Architecture*, Addison-Wesley, 2002.
2. P. Kogut and P. C. Clements, "Feature Analysis of Architecture Description Languages", in *Proceedings of Software Technology Conference*, April 1995.
3. P. C. Clements, "A Survey of Architecture Description Languages", in *Proceedings of the 8th International Workshop on Software Specification and Design*, pp. 16-25, Schloss Velen, Germany, 22-23 March 1996.
4. N. Medvidovic and R. N. Taylor, "A Classification and Comparison Framework for

Software Architecture Description Languages", *IEEE Transactions on Software Engineering*, vol. 26, no. 1, p. 70-93, January 2000.

5. M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik, "Abstractions for Software Architecture and Tools to Support Them", *IEEE Transactions on Software Engineering*, vol. 21, no. 4, pp. 314-335, Apr. 1995.
6. D. Garlan, R. Monroe and D. Wile, "ACME: An Architecture Description Interchange Language", in *Proceedings of CASCON'97*, November 1997.
7. J. E. Robbins, N. Medvidovic, D. F. Redmiles and D. S. Rosenblum, "Integrating Architecture Description Languages with a Standard Design Method", in *Proceedings of the 20th International Conference on Software Engineering*, 1998.
8. N. Medvidovic, D. Rosenblum, D. Redmiles and J. Robbins, "Modelling Software Architectures in the Unified Modeling Language", *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 1, pp. 2-57, January 2002.
9. N. Medvidovic, D. S. Rosenblum and R. N. Taylor, "A Language and Environment for Architecture-Based Software Development and Evolution", in *Proceedings of the 21st International Conference on Software Engineering*, pp. 44-53, May 1999.