Editorial

Towards standardization of adaptable software architectures

Software architecture has been a subject of immense interest for both industry and academia alike recently. Numerous papers have appeared in various conferences highlighting the importance of the architecture for developing usable, reliable, secure and adaptable software systems; books related to architectures have been published and conferences dedicated to software architectures (for example, the IFIP series) are being held. From the industry point of view, we have Bill Gates becoming the Chief Software Architect, more and more companies are creating architecture divisions, and a software architect is becoming a full-time employee in the industry. However, one problem is that the definition of “architecture” itself for software systems is not clear but is going through an evolution process of trial and error. This being the case, it seems extremely difficult at this point in time to “standardize” architecture; what we have in standards such as IEEE-Std-1471-2000: Recommended Practice for Architectural Description of Software-Intensive Systems is a recommended practice for describing architectures.

Adaptability is becoming an attribute that almost every software has to have: IEEE has been trying to institute as part of its standards that every software should be adaptable (for example, IEEE-Std-830-1998: Recommended Practice for Software Requirements Specifications). In order to develop adaptable software, we need an adaptable architecture, since architecture is the first step in the development of software solution. Now comes the question: can we standardize adaptable architectures? Here we have to tackle the even more basic question of what standardization means in software. Concepts such as adaptability and architecture are more conceptual than physical—there can be no one definition for adaptability and neither is there one acceptable definition for architecture. This makes standardizing adaptable architectures extremely difficult. We need an understanding of how to develop adaptable architectures (effectively, adaptable software)—until then standardization will be vacuous. The standardization community will have to come up with new ways to standardize software elements—which includes software architecture as well. There have been some successes in this direction: “reference” architectures are being developed in some domains such as the real-time control system at NIST for the domain of intelligent systems, the TOGAF (The Open Group Architectural Framework) has been proposed for IT systems, “standard” architectures (perhaps, more appropriately, protocol stacks) have been proposed as part of protocols such as TCP/IP, SONET, CDMA, etc. A clear advantage in having standard architectures is that they help in reuse-driven software development. While we do believe that standardization in one form or another may have to come some day, right now, it is somewhat premature in the software engineering community.

The papers that we received for this session on Adaptable Software Architectures highlight some of the problems in developing adaptable software architectures for different domains. In the CFP for this session, we did not have the phrase “selected papers will appear in a journal”; despite this and the fact that the session was not widely advertised we have received more than expected response—we expected six papers but received double that number. This is probably among the first, if not the first, sessions devoted entirely to adaptable software architectures. The domains covered include communications, telepresence, real-time systems, web-based systems, information systems, embedded systems and mobile systems. Availability of standard adaptable architectures for these domains could have helped the re-
searchers and practitioners in these domains a great deal and could have perhaps even allowed for comparisons at the architectural level between the domains.

The session on Adaptable Software Architectures was part of the International Conference on Software Engineering Research and Practice 2002 held at Las Vegas from June 24–27, 2002. The papers in this issue are expanded versions of those presented at the session. The session was well attended and there were about 50 attendees throughout the session.

The first paper by Jyke Jokinen, Hannu-Matti Jarvinen and Tommi Mikkonen entitled “Incremental Introduction of Behaviors with Static Software Architecture” describes the Software Bus Architecture (SBA) that provides a static component structure but permits dynamic behavior modification. This architecture was used to implement a communications device and the experiences are discussed.

The use of commercial off-the-shelf (COTS) components in adaptable software development is an interesting line of research. The next paper by Lawrence Chung, Kendra Cooper, Stephen Lee, Faisal Shafique and Anna Yi entitled “ACASA-A Framework for Adaptable COTS-Aware Software Architecting” discusses the ACASA (Adaptable COTS-Aware Software Architecting) framework that considers adaptability as a goal during the development of COTS-based systems; the use of ACASA in developing a telepresence system is discussed as an example.

One of the important areas of research in the area of adaptable software architectures is the development of mathematical formalisms. The third paper by Masaki Murakami entitled “Evolvable Concurrent Processes” provides a mathematical model of concurrent processes whose functions can be modified by the environment during execution; linear logic has been used to explain this process.

Reflection is being realized as an important requirement for adaptable systems. The fourth paper by Francisco Ortin and Juan Manuel Cueva entitled “Non-Restrictive Computational Reflection” explains the use of reflection in achieving adaptation during run-time; a generic interpreter is used that lets all applications using the interpreter dynamically adapt each other. The process is independent of the programming language used though the illustrations use Python code.

Use of design patterns for developing adaptable software is another promising line of research. The fifth paper by Lawrence Chung, Kendra Cooper and Anna Yi entitled “Developing Adaptable Software Architectures Using Design Patterns: An NFR Approach” illustrates the use of design patterns in developing adaptable software systems—the Proteus framework has been introduced and Proteus’ help in developing adaptable architectures for real-time systems using the example of home appliance control system has been described.

The use of agents in achieving adaptability is explored in the next paper by Patricia Paderewski-Rodriguez, Ma. Jose Rodriguez-Fortiz and Jose Parets-Llorca entitled “An Architecture for Dynamic and Evolving Cooperative Software Agents”. An architectural model to construct cooperative and evolutionary agent-based software systems has been presented—the model has two levels with the higher level allowing changes to the system’s functionality.

Adaptability in web-based systems is emerging as an important topic for research. The seventh paper by Leo Lahav entitled “Hobbes Framework—An adaptable solution for web-driven applications” introduces the Hobbes framework that helps develop adaptable web-based systems; the experiences in using the framework for two practical systems have been described.

Adaptability through reuse is explored in the next paper by Philippe Aniorte entitled “A distributed adaptable software architecture derived from a component model”. This paper describes a component model that uses reuse for developing adaptable software architecture and the model is validated using an information system called ASIMIL.

Automatic generation of adaptable software architectures is explored in the ninth paper by Nary Subramanian and Lawrence Chung entitled “SA³—A Tool for Supporting Adaptable Software Architecture Generation for Embedded Systems”. This paper demonstrates the feasibility of semiautomatic generation of adaptable architectures for the embedded systems domain—the tool uses a systematic approach to search for adaptable components and connections in a knowledge base that can be used by the developer as the starting point for completing the architecture.

The tenth and the last paper by Rick Kazman, Jai Asundi, Jung Soo Kim and Bhuricha Sethananda
entitled “A Simulation Testbed for Mobile Adaptive Architectures” introduces the need for adaptability in mobile systems—this includes the need for graceful degradation of performance, conservation of battery and related issues. The concept of utility as applicable to mobile systems is discussed and the results of simulated studies are presented.

Standardization will certainly help a variety of practitioners. Addressing a variety of topics, we feel that these papers hint at both challenges and opportunities in standardizing software architectures that are adaptable. We further feel that this special issue can be a stepping stone in pursuing further research in this subject matter and a useful body of knowledge to software practitioners.

We thank Prof. Harald Schumny for agreeing to release a special issue on the papers presented at this session; we thank our reviewers Dr. Tommi Mikkonen of Tampere University of Technology, Finland, Dr. Kendra Cooper of University of Texas at Dallas, TX, and Dr. Y.T. Song of Towson University, Maryland, for their fast, efficient and enthusiastic reviewing. We thank all the participants of this session for travelling from different parts of the world and presenting their papers and making the session a successful and enjoyable experience.

Nary Subramanian is employed as Firmware Engineer at Anritsu Company’s Americas Wireless Business Unit at Richardson, TX. He has been involved in implementing man–machine interface firmware for the test and measuring instruments manufactured by Anritsu. He has been working with Anritsu’s Wireless Business Unit for the past 4 years. Anritsu manufactures test and measuring instruments used in the telecom industry. Nary has an MSEE from Louisiana State University, Baton Rouge, LA.

Dr. Lawrence Chung is currently an Associate Professor in the Department of Computer Science at the University of Texas at Dallas. He received his PhD from University of Toronto, and is currently on the Editorial board of Requirements Engineering Journal. He is the principal author of a book, “Non-Functional Requirements in Software Engineering”, which is being adopted in extending object-oriented analysis to goal-oriented and agent-oriented analysis. His research interests include Requirements Engineering, Software Architecture, Electronic Business Systems and Conceptual Modeling.

Nary Subramanian*
Americas Wireless Business Unit, Anritsu Company,
1155 E. Collins Boulevard, Suite 100,
Richardson, TX 75081, USA
E-mail address: narayanan.subramanian@anritsu.com

Lawrence Chung
Department of Computer Science,
University of Texas at Dallas,
Richardson, TX, USA

* Corresponding author. Tel.: +1-469-330-3333; fax: +1-469-330-3310.