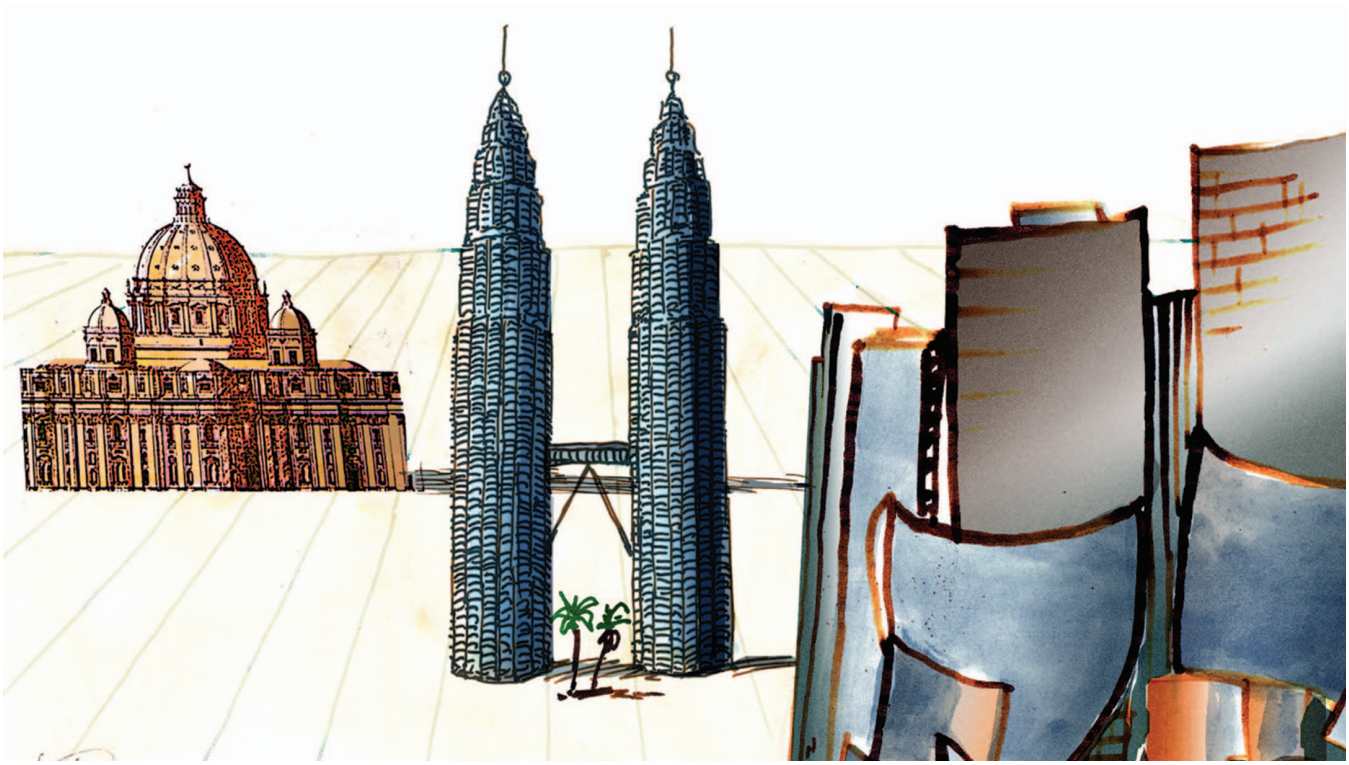
The Past, Present, and Future of Software Architecture

Philippe Kruchten, *University of British Columbia*
Henk Obbink, *Philips Research Europe*
Judith Stafford, *Tufts University*

This special issue celebrates 10 years of software architecture conferences and workshops and the 10th anniversary of the first *IEEE Software* special issue on the topic in November 1995. We aim to address the latest thinking about creating, capturing, and



0740-7459/06/$20.00 © 2006 IEEE

using software architecture throughout a software system's life. The articles we've chosen cover innovative methods and techniques emerging from research to support software architecture practice. They also emphasize the methods, techniques, tools, and software engineering principles that support organizations taking an architecture-centric approach to software development.

What is software architecture?

Software architecture involves

- the structure and organization by which modern system components and subsystems interact to form systems, and
- the properties of systems that can best be designed and analyzed at the system level.

For example, we largely determine end-to-end performance and product-line compatibility by evaluating software architectures. Software architecture captures and preserves designers' intentions about system structure and behavior, thereby providing a defense against design decay as a system ages. It's the key to achieving intellectual control over a sophisticated system's enormous complexity.

Paradoxically, despite the maturing of the discipline, we're far from having a consensus on a satisfying, short, crisp answer to this simple question—no widely accepted definition exists. Paul Clements lists several definitions on the Software Engineering Institute's architecture practice site (www.sei.cmu.edu/architecture/definitions.html). Getting agreement on a definition was the most difficult task in creating the IEEE standard.[1] Actually, this lack of consensus hasn't been a substantial impediment to the discipline's progress, and it regularly provides a source of entertainment at gatherings of software architects.

The software architecture field has many subareas. The International Federation of Information Processing Working Group 2.10 defines these five:

- *Architectural design:* How do we produce an architecture?
- *Analysis:* How do we answer questions, on the basis of an architecture, about certain qualities of the final product?
- *Realization:* How do we produce a system based on an architecture description?
- *Representation:* How do we produce durable artifacts to communicate architecture to humans or machines?
- *Economics:* What architectural issues drive business decisions?

And certainly software architecture is tied closely to other disciplines and communities, such as software design (in general), software reuse, systems engineering and system architecture, enterprise architecture, reverse engineering, requirements engineering, and quality.

A brief history of software architecture

Looking back in time will help us position software architecture's current status and future directions. We also provide pointers to relevant books, papers, conferences, and Web sites.

Pre-1995

The first reference to the phrase *software architecture* occurred in 1969 at a conference on software engineering techniques organized by NATO (see the sidebar "Software Architecture in 1969"). Some of our field's most prestigious pioneers, including Tony Hoare, Edsger Dijkstra, Alan Perlis, Per Brinch Hansen, Friedrich Bauer, and Niklaus Wirth, attended this meeting.

From then until the late 1980s, the word "architecture" was used mostly in the sense of system architecture (meaning a computer system's physical structure) or sometimes in the narrower sense of a given family of computers' instruction set. Key sources about a software system's organization came from Fred Brooks in 1975,[2] Butler Lampson in 1983,[3] David Parnas from 1972 to 1986,[4–7] and John Mills in 1985 (whose article looked more into the process and pragmatics of architecting).[8]

The concept of software architecture as a distinct discipline started to emerge in 1990 (see figure 1). A 1991 article by Winston W. Royce and Walker Royce, father and son, was the first to position software architecture—in both title and perspective—between technology and process.[9] Eberhardt Rechtin dedicated a few sections to software in his 1991 book *Systems Architecting: Creating and Building Complex Systems*.[10] That year, one of us (Philippe Kruchten) wrote an article marrying iterative development with a focus on architecture and defined multiple views for use on a large command-and-control system.[11]

The concept of software architecture as a distinct discipline started to emerge in 1990.

Software Architecture in 1969

Ian P. Sharp made these comments at the 1969 NATO Conference on Software Engineering Techniques. They still resonate well 37 years later.

I think that we have something in addition to software engineering: something that we have talked about in small ways but which should be brought out into the open and have attention focused on it. This is the subject of software architecture. Architecture is different from engineering.

As an example of what I mean, take a look at OS/360. Parts of OS/360 are extremely well coded. Parts of OS, if you go into it in detail, have used all the techniques and all the ideas which we have agreed are good programming practice. The reason that OS is an amorphous lump of program is that it had no architect. Its design was delegated to a series of groups of engineers, each of whom had to invent their own architecture. And when these lumps were nailed together they did not produce a smooth and beautiful piece of software.

I believe that a lot of what we construe as being theory and practice is in fact architecture and engineering; you can have theoretical or practical architects: you can have theoretical or practical engineers. I don't believe, for instance, that the majority of what Dijkstra does is theory—I believe that in time we will probably refer to the "Dijkstra School of Architecture."

What happens is that specifications of software are regarded as functional specifications. We only talk about what it is we want the program to do. It is my belief that anybody who is responsible for the implementation of a piece of software must specify more than this. He must specify the design, the form; and within that framework programmers or engineers must create something. No engineer or programmer, no programming tools, are going to help us, or help the software business, to make up for a lousy design. Control, management, education and all the other goodies that we have talked about are important; but the implementation people must understand what the architect had in mind.

Probably a lot of people have experience of seeing good software, an individual piece of software which is good. And if you examine why it is good, you will probably find that the designer, who may or may not have been the implementer as well, fully understood what he wanted to do and he created the shape. Some of the people who can create shape can't implement and the reverse is equally true. The trouble is that in industry, particularly in the large manufacturing empires, little or no regard is being paid to architecture. —Software Engineering Techniques: Report of a Conference Sponsored by the NATO Science Committee, *B. Randell and J.N. Buxton, eds., Scientific Affairs Division, NATO, 1970, p. 12.*

In 1992, Dewayne Perry and Alexander Wolf published their seminal article "Foundations for the Study of Software Architecture."[12] This article introduced the famous formula "{elements, forms, rationale} = software architecture," to which Barry Boehm added "constraints" shortly thereafter. For many researchers, the "elements" in the formula were components and connectors. These were the basis for a flurry of architecture description languages (ADLs), including C2, Rapide, Darwin, Wright, ACME, and Unicon, which unfortunately haven't yet taken much root in industry.

1994 saw the first book on software architecture, by former IBMers Bernard Witt, F. Terry Baker, and Everett Merrit.[13]

1995–1998

In 1995, software architecture really started to bloom, and events accelerated with numerous contributions to the field from industry and academia. Notable examples were the Software Architecture Analysis Method (SAAM), the first of a Software Engineering Institute series of methods;[14] several approaches involving multiple views such as Rational's 4+1 views[15] or Siemens' four views;[16] and specific design patterns for software architecture.[17] Siemens,[18] Nokia,[19] Philips,[20] Nortel, Lockheed Martin, IBM, and other large software development organizations—mainly in systems, aerospace, and telecommunications—started to pay attention to software architecture, teaming up with the reuse community to investigate software product line architectures.[21] Another book by Rechtin and Mark Maier, *The Art of Systems Architecting*, nicely filled the gap between system and software.[22]

1999–2005

1999 was another key year for software architecture, seeing the first IFIP Conference on Software Architecture[23] and the founding of IFIP Working Group 2.10 and the Worldwide Institute of Software Architects. Many nonacademics started to pitch in best practices.[24–27] In hopes of increasing the practice of architecture description, the Open Group introduced the Architecture Description Markup Language, an XML-based ADL that provides support for broad sharing of architectural models. Joining forces with the reuse and product-families communities, software product lines became a sort of subdiscipline, attracting lots of attention of large manufacturing companies. New methods such as SAAM, BAPO, and ATAM emerged or consolidated.[14,28,29] We had one general architecture standard, RM-ODP,[30,31] and we added one, IEEE 1471.[1] The SEI team produced book after book,[29,32–34] and more.

Where are we now?

Large companies such as Microsoft have

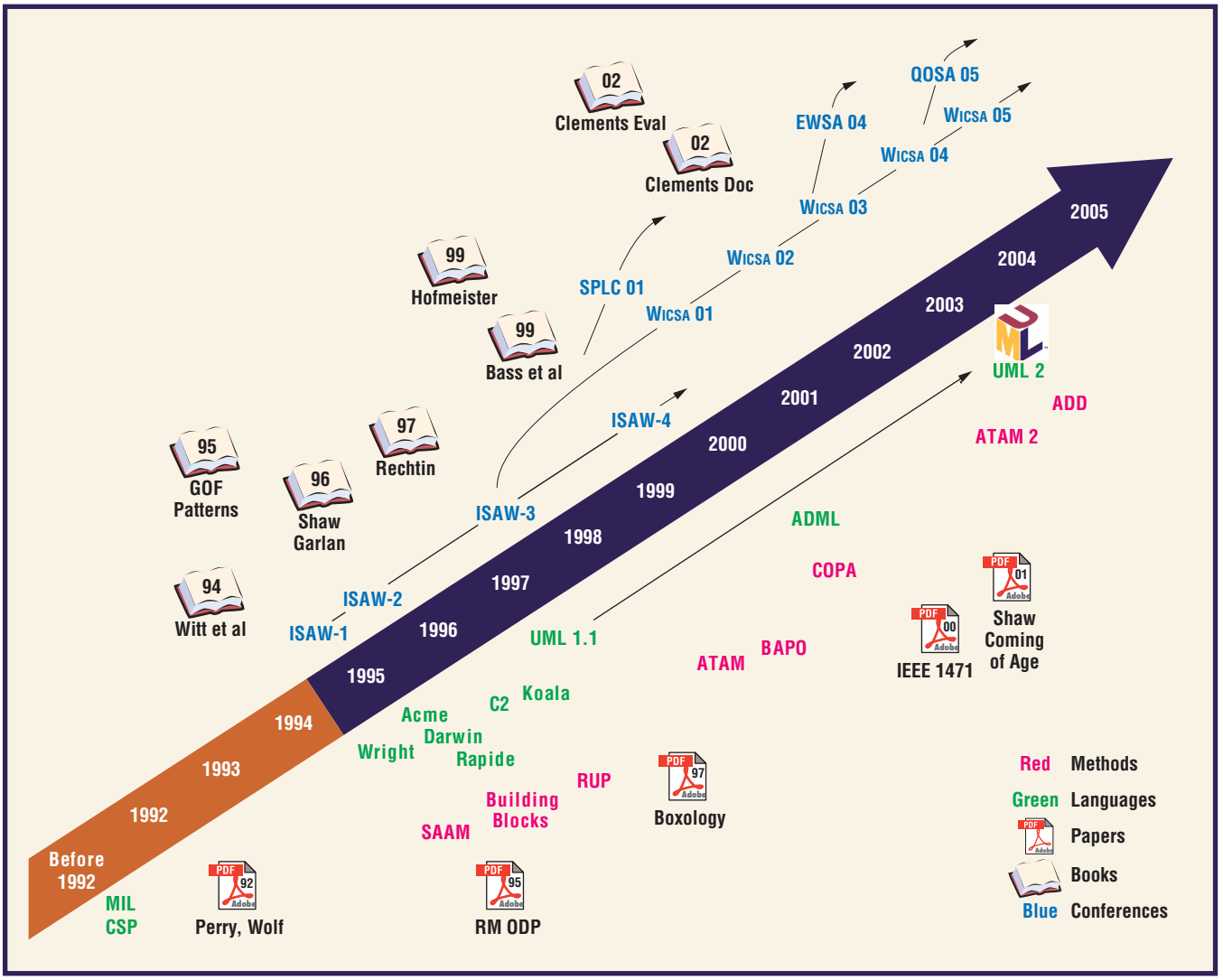


Figure 1. Ten years of software architecture.

their chief architects. There's been a slight inflation in titles from software designer and developer to software architect, despite Mary Shaw's plea a few years ago to avoid calling everything in sight architecture.

We now have many rich ADLs. But few are in practical use except perhaps Koala,[35] and perhaps UML if you consider it an ADL (many ADL purists do not).

For several domains, precooked architectures exist in the form of platforms—for instance, J2EE, .NET, Symbian/Series 60, and Websphere. Application layer interchange standards such as XML and SOAP have a significant impact on these architectures. Scripting languages like Python and Perl change the way we construct systems. Architects can't start from scratch anymore; they build systems based on their insights regarding these platforms' capabilities. Also, open source software is strongly affecting the practice.

A body of software architecture knowledge is readily accessible in more than 25 books (see the "Architecture Library" sidebar) and numerous articles (see the "Great Papers" sidebar). Dozens of universities around the world teach software architecture, many organizations offer architect training courses, and an active community has formed (see the "Software Architecture Community" sidebar).

A discipline has emerged.

The articles in this issue

The importance of software architecture for software development is widely recognized, yet transfer of innovative techniques and methods from research to practice is slow. We chose the articles in this issue specifically

Starting Your Software Architecture Library

We recommend the following 12 books to any budding software architect. They cover a vast range of issues and provide the necessary foundation for further study, research, and application.

The first book

- M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996. This book put software architecture firmly on the world map as a discipline distinct from software design or programming, and it's still a worthwhile read. The authors tried to define what software architecture is—a difficult task. We still haven't reached consensus 10 years later. Much of the book is dedicated to the concept of architectural styles, and there's a useful chapter on educating software architects.

The SEI trilogy

- L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed., Addison-Wesley, 2003. Originally published in 1998, this book expanded many aspects of software architecture: process and method, representation, techniques, tools, and business implications. It provides a good introduction to several SEI architectural methods.
- P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, 2002. Focused solely on software architecture documentation and representation, this book constitutes a de facto application guide to the rather abstract *IEEE Standard 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems*.
- P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architecture*, Addison-Wesley, 2002. How good is this architecture? The third book in the SEI trilogy (this productive group actually wrote more than three) focuses on reviewing and evaluating various aspects of "goodness" and qualities of an architecture, existing or to be built. A good complement to the *Software Architecture Review and Assessment (SARA) Report* (SARA Working Group, 2002).

Ammunition for architects

- C. Hofmeister, R. Nord, and D. Soni, *Applied Software Architecture*, Addison-Wesley, 1999. The authors offer a systematic, detailed architectural-design method and a representation of software architecture based on their work at the Siemens Research Center.
- I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success*, Addison-Wesley, 1997. As the title indicates, this book bridges the software reuse community (which was thriving but running a bit out of air in the mid-1990s) with the architecture community, showing how the two can leverage each other. It presents elements of the architectural method the Rational Unified Process embodies. If reuse and product lines are important to you, we'll suggest more reading later.
- F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons, 1996. Branching off from the design pattern work of the Gang of Four, this "Gang of Five" assembled a useful catalog of architectural-design patterns. Unfortunately, they haven't continued what they had started so well.

Pragmatics

- R.C. Malveau and T.J. Mowbray, *Software Architect Bootcamp*, 2nd ed., Prentice Hall, 2000. This "how to get started" guide is directed at practitioners.
- D.M. Dikel, D. Kane, and J.R. Wilson, *Software Architecture: Organizational Principles and Patterns*, Prentice Hall, 2001. The authors have captured the dynamics of what happens in a software architecture team—the constraints, tensions, and dilemmas—in their VRAPS (vision, rhythm, anticipation, partnering, and simplification) model.
- E. Rechtin and M. Maier, *The Art of Systems Architecting*, CRC Books, 1997. Rechtin's initial book in 1991 was about systems and very little about software, although software architects could transpose and interpret many of the principles presented. By teaming up with Mark Maier, Rechtin was able to cover software aspects more specifically and deeply. However, beginners will find it hard to read, so they should start with the first two books in this group.

Software product lines

- J. Bosch, *Design and Use of Software Architecture: Adopting and Evolving a Product-Line Approach*, Addison-Wesley, 2000. This book and the next represent the branching off of software architecture into its application for software product lines.
- M. Jazayeri, A. Ran, F. van der Linden, and P. van der Linden, *Software Architecture for Product Families: Principles and Practice*, Addison-Wesley, 2000.

because their ideas and results will be of interest to readers wishing to increase the adoption of software architecture-related techniques and methods in their work environment. "The Golden Age of Software Architecture" by Mary Shaw and Paul Clements specifically addresses the discipline's maturation. After applying the Redwine-Riddle model of technology maturation to this field, the authors conclude that the broad concept of software architecture has

Great Papers on Software Architecture

If you're not a great fan of books (see the "Starting Your Software Architecture Library" sidebar), you can get a quick introduction to many of the underlying concepts from this collection of key papers.

Foundations

- M. Shaw and D. Garlan, "An Introduction to Software Architecture," V. Ambriola and G. Tortora, eds., *Advances in Software Engineering and Knowledge Engineering*, vol. 2, World Scientific Publishing, 1993, pp. 1–39. Shortly preceding their book, this paper brought together what we knew about software architecture in the beginning of the 1990s.
- D.E. Perry and A.L. Wolf, "Foundations for the Study of Software Architecture," *ACM Software Eng. Notes*, vol. 17, no. 4, 1992, pp. 40–52. This seminal paper will be always remembered for giving us this simple but insightful formula: {elements, form, rationale} = software architecture.

Precursors

- D.L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Comm. ACM*, vol. 15, no. 12, 1972, pp. 1053–1058. Software architecture didn't pop up out of the blue in the early 1990s. Although David Parnas didn't use the term "architecture," many of the underlying concepts and ideas owe much to his work. This article and the next two are the most relevant in this regard.
- D.L. Parnas, "On the Design and Development of Program Families," *IEEE Trans. Software Eng.*, vol. 2, no. 1, 1976, pp. 1–9.
- D.L. Parnas, P. Clements, and D.M. Weiss, "The Modular Structure of Complex Systems," *IEEE Trans. Software Eng.*, vol. 11, no. 3, 1985, pp. 259–266.
- F. DeRemer and H. Kron, "Programming-in-the-Large versus Programming-in-the-Small," *Proc. Int'l Conf. Reliable Software*, ACM Press, 1975, pp. 114–121. Their Module Interconnection Language (MIL 75) is in effect the ancestor of all ADLs, and its design objectives are still valid today. The authors had a clear view of architecture as distinct from design and programming at the module level but also at the fuzzy, abstract, "high-level design" level.

Architectural views

- D. Soni, R. Nord, and C. Hofmeister, "Software Architecture in Industrial Applications," *Proc. 17th Int'l Conf. Software Eng.* (ICSE 95), ACM Press, 1995, pp. 196–207. This article introduced Siemens' five-view model, which the authors detailed in their 1999 book *Applied Software Architecture* (see the "Architecture Library" sidebar).
- P. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, vol. 12, no. 6, 1995, pp. 45–50. Part of the Rational Approach—now known as the Rational Unified Process—this set of views was used by many Rational consultants on large industrial projects. Its roots are in the work done at Alcatel and Philips in the late 1980s.

Process and pragmatics

- B.W. Lampson, "Hints for Computer System Design," *Operating Systems Rev.*, vol. 15, no. 5, 1983, pp. 33–48; reprinted in *IEEE Software*, vol. 1, no. 1, 1984, pp. 11–28. This article and the next gave one of us (Kruchten), a budding software architect in the 1980s, great inspiration. They haven't aged and are still relevant.
- J.A. Mills, "A Pragmatic View of the System Architect," *Comm. ACM*, vol. 28, no. 7, 1985, pp. 708–717.
- W.E. Royce and W. Royce, "Software Architecture: Integrating Process and Technology," *TRW Quest*, vol. 14, no. 1, 1991, pp. 2–15. This article articulates the connection between architecture and process very well—in particular, the need for an iterative process in which early iterations build and validate an architecture.

Two more for the road

Where do we stop? We're tempted to add many more articles on such ADLs as Rapide, Wright, and C2 as well as on model-driven architecture. We'll just add two more.

- M. Shaw and P. Clements, "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems," *Proc. 21st Int'l Computer Software and Applications Conf.* (COMPSAC 97), IEEE CS Press, 1997, pp. 6–13.
- M. Shaw, "The Coming-of-Age of Software Architecture Research," *Proc. 23rd Int'l Conf. Software Eng.* (ICSE 01), IEEE CS Press, 2001, pp. 656–664a.

run the full course of the model and is a fully mature technology. This article updates Mary Shaw's 2001 article "The Coming-of-Age of Software Architecture Research,"[36] which remains a must-read for anyone who wants to embark on software architecture research.

The other articles in this issue address a wide range of topics targeted at the software development community as a whole. We selected these articles from among three dozen submissions to highlight the field's current trends.

The first article, "In Practice: UML Software Architecture and Design Description" by Christian Lange, Michel Chaudron, and Johan

A Software Architecture Community

Here are a dozen places to go for more information on software architecture, to participate in or attend conferences, or to join a group of peers.

Resources

- The Software Engineering Institute's Software Architecture for Software-Intensive Systems Web site (www.sei.cmu.edu/architecture) contains many definitions, papers on their methods, and further pointers. The software architecture practice group at the Software Engineering Institute maintains this portal.
- The Gaudí System Architecting Web page (www.gaudisite.nl), named after the famous Spanish architect, deals with system architecture. Gerrit Muller from Philips Research and the Embedded Systems Institute in Eindhoven maintains the page.
- The Bredemeyer architecture portal (www.bredemeyer.com), called "Software Architecture, Architects and Architecting," is maintained by Dana Bredemeyer and Ruth Malan. It contains not only their own writings but also a well-organized collection of other resources and announcements.
- The Software Product Lines page (http://softwareproductlines.com) focuses on product lines and large-scale reuse.
- SoftwareArchitectures.com (http://softwarearchitectures.com) is another portal to architecture resources.
- Grady Booch, from IBM, is spearheading an effort to establish a handbook for software architects and is building a repository of example architectures and case studies (www.booch.com/architecture/index.jsp).

Conferences

- Working IEEE/IFIP Conferences on Software Architecture (www.softwarearchitectureportal.org/WICSA/conferences). Since 1999, WICSA has attracted a lot of contributions from industry and academia and many interesting debates and advances. Its location alternates between North America and another part of the world (only Europe so far). WICSA subsumed the International Software Architecture Workshop series which ran from 1995 to 2000.
- European Workshop on Software Architecture (www.arch-ware.org/ewsa). Begun in 2004, EWSA is mainly driven by the participants in the European project ArchWare—Architecting Evolvable Software.
- Software Product Line Conference (http://softwareproductlines.com). Since 2000, this subcommunity of software architecture has organized a successful meeting series. SPLC subsumed the older PFE (Product Family Engineering) conference series in Europe. Start from this site to access the most recent SPLC.
- The Conference on the Quality of Software Architectures, or QOSA (http://se.informatik.uni-oldenburg.de/qosa) was a new conference in 2005.
- Software architecture is also present—often in the form of a specific session or a distinct track—in other conferences, including ICSE; ECOOP (European Conference on Object-Oriented Programming); OOPSLA (Object-Oriented Programming Systems, Languages, and Applications); FSE (Foundation of Software Engineering); APSEC (Asia-Pacific Software Engineering Conference); and now MODELS (ACM/IEEE International Conference on Model-Driven Engineering Languages and Systems), which subsumed the UML conference series.

Associations and working groups

- IFIP WG 2.10 Software Architecture (www.ifip.org/bulletin/bulltcs/memtc02.htm#wg210). Founded at the first WICSA conference in 1999, the 13 or so members of the IFIP's Working Group 2.10 meet face to face twice a year and a few more times by telephone and the Internet. They are the driving force behind the WICSA conference series and this special issue of *IEEE Software*. They also maintain the www.softwarearchitectureportal.org portal.
- The Worldwide Institute of Software Architects, or WWISA (www.wwisa.org) was founded by Mark and Laura Sewell in 1999.
- The International Association of Software Architects, or IASA (www.iasarchitects.org) is an association of IT architects focusing on social networking, advocacy, ethics, and knowledge sharing.
- IEEE Standards Association WG 1471 (http://standards.ieee.org). The working group that created *IEEE Std 1471-2000* is now resurrecting itself to tackle a revision of the standard.
- SARA—Software Architecture Review and Assessment. This informal group of architects from industry (Philips, Siemens, Rational, Nokia, IBM, and Lockheed Martin) met regularly from 1998 to 2001 to share software evaluation practices. They produced a report in 2001 (www.philippe.kruchten.com/architecture/SARAv1.pdf).
- Research and education. Many academics and researchers maintain pages with pointers to their research and other resources. We picked just two examples: Nenad Medvidovic, University of Southern California, http://sunset.usc.edu/research/software_architecture/index.html; and Gert Florijn, Software Engineering Research Center in the Netherlands, www.serc.nl/people/florijn/interests/arch.html.

Muskens provides a large-scale survey of UML use and its associated benefits and problems. The second article, "Software Architecture-Centric Methods and Agile Development" by

Robert L. Nord and James E. Tomayko (recently deceased), presents the interesting concept of leveraging the best aspects of architecture-centric methods and agile development. Combining these approaches can help address quality attributes and maintain flexibility. Michael Stal's article, "Using Architectural Patterns and Blueprints for Service-Oriented Architecture," uses published patterns to explain an SOA-based system's fundamental structures. This subject area interests most practitioners and is particularly relevant at the moment.

"Using Architecture Models for Runtime Adaptability" by Jacqueline Floch, Svein Hallsteinsen, Erlend Stav, Frank Eliassen, Ketil Lund, and Eli Gjørven extends the need for architectural reasoning to runtime analysis, because of the advent of software platforms that support component plug-ins and dynamic binding. It shows that you can achieve a dynamically adaptive architecture by following a well-defined approach that cleanly separates various concerns. It also shows an interesting use of architecture models that are available at runtime and takes architecture in new directions, including mobile computing and runtime adaptability. The fifth article, "Architecture Description Languages for High-Integrity Real-Time Systems" by Alek Radjenovic and Richard Paige, addresses architectural concerns beyond the typical structural ones and the use of an ADL for that purpose. Finally, "A Fault-Tolerant Architectural Approach for Dependable Systems" by Rogério de Lemos, Paulo Asterio de Castro Guerra, and Cecília Mary Fischer Rubira describes the process ingredients to arrive at a dependable system, sketches the role of exception handling in fault tolerance, and introduces an ideal fault-tolerant architectural element.

About the Authors

Philippe Kruchten is a professor of software engineering at the University of British Columbia. His research interests include software architecture, software development processes, and software project management. He received his doctorate degree from L'École Nationale Supérieure des Télécommunications. He's a founding member of IFIP WG2.10 on Software Architecture. He's a professional engineer in British Columbia, a Certified Software Development Professional, a senior member of the IEEE Computer Society, and the author of *The Rational Unified Process—An Introduction* (Addison-Wesley, 2003, 3rd ed.). Contact him at the UBC, 2332 Main Mall, Vancouver, BC V6T1Z4, Canada; kruchten@ieee.org.

Henk Obbink is a principal scientist at Philips Research Laboratories in Eindhoven. He heads the architecture research team for software-intensive healthcare systems. His research interests have included computer systems, communication systems, defense systems, and consumer products. He received his doctorate in chemistry and physics from the University in Utrecht. He's a member of the IFIP Working Group 2.10 on Software Architecture and the steering committees of the Working IEEE/IFIP Conference on Software Architecture and the Software Product Line Conference. Contact him at Philips Research Laboratories Europe, High Tech Campus 31, WDC 2.030, 5656 AE Eindhoven, the Netherlands; henk.obbink@philips.com.

Judith Stafford is a senior lecturer at Tufts University and a visiting scientist at Carnegie Mellon University's Software Engineering Institute. Her research focuses on software architecture analysis, architecture support for software component composition, and software architecture documentation. She coauthored *Documentation Software Architectures* (Addison-Wesley, 2002). She received her PhD in computer science from the University of Colorado at Boulder. She's a member of the IEEE Computer Society, ACM SIGSOFT and SIGPLAN, and the IFIP Working Group on Software Architecture (WG2.10). Contact her at the Dept. of Computer Science, Tufts Univ., Medford, MA 02155; jas@cs.tufts.edu; www.cs.tufts.edu/~jas.

What will the next decade bring? A lot of research is dedicated to model-driven architecture, again with UML in the background,[37,38] while research on special-purpose ADLs seems to have reached a plateau. Some researchers are looking into architectural knowledge—that is, architectural-design decisions and their rationale.[39] Increasing support for educating software architects will include more and better case studies and textbooks. Tools that are specifically tailored for architects will be made available to support their design, representation, analysis, and implementation tasks. The concept of aspects will intersect with and affect research on the quality of software architecture. Software architecture will be recognized as a key foundation to agile software development, despite the fact that it's ignored or even despised by the most ardent "agilistas," who have nicknamed it BUFD (big up-front design).

Acknowledgments

We thank our colleagues in IFIP Working Group 2.10 on Software Architecture for their input, particularly those who participated in the discussions at our August 2005 meeting in Krapi, Finland.

References

1. *IEEE 1471:2000, Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Press, 2000.
2. F.P. Brooks Jr., *The Mythical Man-Month*, Addison-Wesley, 1975.
3. B.W. Lampson, "Hints for Computer System Design," *Operating Systems Rev.*, vol. 15, no. 5, 1983, pp. 33–48.
4. D.L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Comm. ACM*, vol. 15, no. 12, 1972, pp. 1053–1058.
5. D.L. Parnas, "On the Design and Development of Program Families," *IEEE Trans. Software Eng.*, vol. 2, no. 1, 1976, pp. 1–9.
6. D.L. Parnas and P. Clements, "A Rational Design

Process: How and Why to Fake It," *IEEE Trans. Software Eng.*, vol. 12, no. 2, 1986, pp. 251–257.
7. D.L. Parnas, P. Clements, and D.M. Weiss, "The Modular Structure of Complex Systems," *IEEE Trans. Software Eng.*, vol. 11, no. 3, 1985, pp. 259–266.
8. J.A. Mills, "A Pragmatic View of the System Architect," *Comm. ACM*, vol. 28, no. 7, 1985, pp. 708–717.
9. W.E. Royce and W. Royce, "Software Architecture: Integrating Process and Technology," *TRW Quest*, vol. 14, no. 1, 1991, pp. 2–15.
10. E. Rechtin, *Systems Architecting: Creating and Building Complex Systems*, Prentice Hall, 1991.
11. P. Kruchten, "Un Processus de Développement de Logiciel Itératif et Centré sur l'Architecture [An Iterative Software Development Process Centered on Architecture]," *Proc. 4ème Congrès de Génie Logiciel*, EC2, 1991, pp. 369–378.
12. D.E. Perry and A.L. Wolf, "Foundations for the Study of Software Architecture," *ACM Software Eng. Notes*, vol. 17, no. 4, 1992, pp. 40–52.
13. B. Witt, F.T. Baker, and E. Merritt, *Software Architecture and Design: Principles, Models, and Methods*, Van Nostrand Reinhold, 1994.
14. R. Kazman et al., "SAAM: A Method for Analyzing the Properties of Software Architectures," *Proc. 16th Int'l Conf. Software Eng.* (ICSE 94), IEEE CS Press, 1994, pp. 81–90.
15. P. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, vol. 12, no. 6, 1995, pp. 45–50.
16. D. Soni, R. Nord, and C. Hofmeister, "Software Architecture in Industrial Applications," *Proc. 17th Int'l Conf. Software Eng.* (ICSE-17), ACM Press, 1995, pp. 196–207.
17. F. Buschmann et al., *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons, 1996.
18. C. Hofmeister, R. Nord, and D. Soni, *Applied Software Architecture*, Addison-Wesley, 1999.
19. A. Ran, "ARES Conceptual Framework for Software Architecture," *Software Architecture for Product Families: Principles and Practice*, M. Jazayeri, A. Ran, and F. van der Linden, eds., Addison-Wesley, 2000, pp. 1–29.
20. J.K. Müller, "Integrating Architectural Design into the Development Process," *Proc. 1995 Int'l Symp. and Workshop Systems Eng. of Computer-Based Systems*, IEEE Press, 1995, pp. 114–121.
21. I. Jacobson, K. Palmkvist, and S. Dyrhage, "Systems of Interconnected Systems," *Report on Object-Oriented Analysis and Design* (*ROAD*), vol. 2, no. 1, May-June 1995.
22. E. Rechtin and M. Maier, *The Art of Systems Architecting*, CRC Books, 1997.
23. P. Donohue, ed., *Software Architecture—1st IFIP Conf. Software Architecture* (WICSA 1), Kluwer Academic Publishers, 1999.
24. R.C. Malveau and T.J. Mowbray, *Software Architect Bootcamp*, 2nd ed., Prentice Hall, 2000.
25. D.M. Dikel, D. Kane, and J.R. Wilson, *Software Architecture: Organizational Principles and Patterns*, Prentice Hall, 2001.
26. H. Obbink et al., *Report on Software Architecture Review and Assessment (SARA)*, V1.0, Feb. 2002; www.philippe.kruchten.com/architecture/SARAv1.pdf.
27. P. Kruchten, *The Rational Unified Process—An Introduction*, Addison-Wesley, 1998.
28. H. Obbink et al., "COPA: A Component-Oriented Platform Architecting Method for Families of Software-Intensive Electronic Products (Tutorial)," *Proc. 1st Software Product Line Conf.* (SPLC1), 2000; www.extra.research.philips.com/SAE/COPA/COPA_Tutorial.pdf.
29. P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architecture*, Addison-Wesley, 2002.
30. *ISO/IEC 10746:1995, Reference Model of Open Distributed Processing (RM-ODP)*, ITU Rec. X901, 1995.
31. J. Putman, *Architecting with RM-ODP*, Prentice Hall, 2000.
32. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.
33. P. Clements et al., *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, 2002.
34. P. Clements and L. Northrop, *Software Product Lines: Practice and Patterns*, Addison-Wesley, 2002.
35. R. van Ommering et al., "The Koala Component Model for Consumer Electronics," *IEEE Trans. Computers*, vol. 33, no. 3, 2000, pp. 78–85.
36. M. Shaw, "The Coming-of-Age of Software Architecture Research," *Proc. 23rd Int'l Conf. Software Eng.* (ICSE 01), IEEE CS Press, 2001, pp. 656–664a.
37. B. Selic, "The Pragmatics of Model-Driven Development," *IEEE Software*, vol. 20, no. 5, 2004, pp. 19–25.
38. R. Soley, *Model-Driven Architecture*, Object Management Group, 2000.
39. J. Bosch, "Software Architecture: The Next Step," *Proc. 1st European Workshop Software Architecture* (EWSA 04), Springer, 2004, pp. 194–199.