Editor: Warren Keuffel ■ wkeuffel@computer.org

Software Architecture: A Maturing Discipline

Robert C. Larrabee

Design & Use of Software Architectures: Adopting and Evolving a Product-Line Approach, *by Jan Bosch, Addison-Wesley/ACM Press, Reading, Mass., 2000, ISBN 0-201-67494-7, 354 pp., US$49.95.*

I spend a lot of time talking and simultaneously smudging up dry-erase boards. Much of this time, I'm drawing software architectures of one flavor or another (there appear to be more than "4+1" flavors possible, as David Parnas would likely agree). The abstractions inherent to this field seem to require tangible representation, just so listeners can follow the ideas. Looking at the audience as I speak, I often see them begin to understand a concept when I draw, but I see this happen less often when I just talk. Software architecture is a relatively new discipline, and it doesn't seem to have settled into our collective psyche quite yet.

Software architecture doesn't seem to have settled into our collective psyche.

Design & Use of Software Architectures is a valuable contribution to the still-maturing field of software architecture and component development. Recent similar works have defined software architecture (*The Art of Systems Architecting* by Mark Maier and Eberhardt Rechtin), addressed it in a business or technical context (*Software Architecture in Practice* by Len Bass, Paul Clements, and Rick Kazman), and introduced the Zen of it (*Software Architect Bootcamp* by Raphael Malveau and Thomas Mowbray). The IEEE recently published its architectural standard 1471 titled *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Many other current works address component development "in the small" (Corba and DCOM, for instance), whereas this book tackles component development and reuse at the macro level. It elaborates on the excellent treatment of component-based development and product-line practice given in *Software Architecture in Practice*.

Meta architectures

Just as there are books about writing books (*The Chicago Manual of Style*) and processes for developing processes (IDEF0), there are designs for architectures (*Design & Use of Software Architectures*). In order for this emerging discipline to be accepted as true engineering, it should have some formality. Jan Bosch offers three architectural design case studies, similar to but less detailed than the A-7 aircraft avionics architectural case study in *Software Architecture in Practice*.

The author suggests that architectural-design patterns exist and, if you can design a software architecture, then perhaps you can also identify its reusable constituents, just as assembly-language programmers once did. These constituents were called macros back then; now "components" has a similar meaning. If you can identify and appropriately design these reusable components, you do not need to recreate them.

Many depictions—drawings, graphical and textual representations, hyperlinked models, and so on—are passed off as software architecture. How can we determine whether an "architecture" is indeed sound, or rather a hasty response to a deliverable re-

0740-7459/02/$17.00 © 2002 IEEE

quirement? Bosch provides effective heuristics to ascertain this as well as an effective weighted-analysis method (which is essentially a Quality Function Deployment matrix, or multicriteria decision-making methodology). He also develops and lists various assessment criteria, including the Software Engineering Institute's Software Architectural Analysis Method. I was disappointed, though, that Bosch did not build on the SEI's Architectural Tradeoff Analysis Method, which is a logical extension of the SAAM and employed to refine architecture.

Business benefits of product-line architectures

Product-line practice is a parallel emerging discipline, and the SEI has an entire group devoted to exploring its potential (Clements and Bass are associated with it). Product-line practices and architectures are clearly connected, and Bosch thoroughly explores the possible synergies. He devotes about half of the book to these concepts, describing how to design a product-line architecture and developing the business case benefits for doing so. He again addresses reuse, but from the business perspective (if you build it once…). The simplistic Lego approach (everything will fit the universal interface) is addressed and dissected here, and the idea of architectural components is developed and presented particularly well.

Are we there yet?

Well, no, we are not, but this book helps. The content is good and complements the previously mentioned texts and the new IEEE standard, though it is sparse on graphics (unfortunately, as most software architectural books are). Also, there are many typographical errors, which tend to slow down the careful reader. The intended audience includes technical managers and system architects, but the material is not overly academic. The author's case studies are particularly instructive to anyone contemplating the business benefits of product-line architectures.

Robert C. Larrabee is a senior principal staff engineer at Arinc Research. His interests are in software-entropy reduction and the beauty of abstraction. Contact him at larrabeerc@navair.navy.mil.

A Gentle Introduction to Embedded Systems

Diomidis Spinellis

Microcontroller Projects with Basic Stamps, *by Al Williams, R&D Books, Gilroy, Calif., 2000, ISBN 0-87930-587-8, 407 pp., US$44.95.*

Information appliances and ubiquitous computing are the wave of the future, yet many of us are designing and implementing software for desktop or server machines with hundreds of Mbytes of RAM running at hundreds of MHz, thereby drawing hundreds of watts. To exploit the power of our machines, we run complex operating systems, programming languages, development tools, and communication protocols on them. *Microcontroller Projects with Basic Stamps* introduces a different world, where program listings fit on a single page, memory resources are measured in bytes, and self-contained designs communicate directly with the outside world using switches, lights, and motors.

A simple hardware platform

Anyone who has contemplated the design of a microprocessor-based prototype in the last 10 years will have come across a number of obstacles. To decrease size and increase communication bandwidth, modern microprocessors use hundreds of pins and surface-mounted technology. This pushes their use beyond the soldering-iron hobbyist's domain. Memory and I/O interfacing are also more complicated: modern processors are optimized for buses with intelligent arbitration protocols and multilevel caches. However, hardware evolution has also benefited microcontroller-based devices. The "Basic Stamps" that Al Williams presents are self-contained microcontrollers. They have eight or 16 (depending on the product version) readily accessible I/O pins and can run for hours on a single 9V battery. They can be directly programmed in a variation of Basic using an editor running in Microsoft Windows. Developers can then download the programs into the Stamp's nonvolatile memory using a cable attached to the PC. From that point on, the Stamp will function as a stand-alone appliance.

Appliances communicate with the outside world using sensors and simple

output devices such as speakers, actuators, and liquid crystal displays. Fortunately, designers with mostly software experience will be able to experiment with their own projects after learning from the book's clear descriptions of the practical aspects that underlie hardware implementations. Readers will learn how to use Ohm's law to calculate the value of pull-up resistors to drive LEDs, how to amplify sound using operational amplifiers, and how to reverse-engineer a stepper motor taken from an old floppy drive and give it new life as a robot part. For those elements not covered in this book, I recommend the classic *The Art of Electronics* (Paul Horowitz and Winfield Hill, University Press, 2nd ed., 1989) as a companion book.

Apart from a large reference chapter describing the Stamp's Basic dialect, most of the book is presented in the form of concrete, self-contained projects—a capacitance meter, a PC-based frequency counter, a phone dialer, a reaction game, a pocket watch, a Morse-code keyer. A circuit diagram and a listing of the firmware accompany each project. The chapters are devoted to digital and analog I/O, serial communications, LCDs and keypads, and motors. More adventurous readers might wish to experiment with PIC, the microprocessor that forms the Basic Stamp's core; a separate chapter provides introductory information.

Minor problems, but an inspiring book

A few minor problems distract from an otherwise lively, well-written book. I missed a formal reference covering the Stamp's hardware and found the gray background used for the examples an unfortunate choice. However, the complete Stamp reference and all the examples are included in the accompanying CD-ROM. In addition, the chapter covering the Basic commands would better fit in an appendix; a running header that is wrong for the chapter's 100 pages hinders its casual browsing.

Overall, the book left me with an urge to order hardware and try out some Stamp projects. I recommend it as a starter for those wanting to spice up their software designs with some hardware.

The author's approach, to design a case study and then apply it, is good, plus you can download the sources.

Diomidis Spinellis is an assistant professor in the Department of Management Science and Technology at the Athens University of Economics and Business in Athens. Contact him at dds@aueb.gr.

Practical Transactional Systems in the Internet Age

Angelo Bellotti

Enterprise Transaction Processing Systems: Putting the CORBA OTS, Encina++ and OrbixOTM to Work, *by Ian Gorton, Addison-Wesley, Reading, Mass., 2000, ISBN 0-201-39859-1, 240 pp., US$49.99.*

This book is a practical approach to the complex and evolving world of distributed applications. It introduces readers to the characteristics and complexities of typical transactional systems, which represent the main infrastructure and logistical support of every medium-size and big company. Nevertheless, you can apply a number of the book's concepts to other systems.

Ian Gorton focuses on transactional-systems design and development using cutting-edge Corba technology and in particular its Object Transaction Service. The book is not about Corba, but its overview of the technology will be useful for beginners. Two examples are developed with available products, Encina++ and OrbixOTM. You're lucky if you are using one of those products.

The first part of the book is an excellent introduction to transactional systems; I have rarely found such a clear, useful presentation. Next, the author thoroughly describes real-world distributed systems and transactional issues. Making extensive use of UML notation, he gives a good explanation of Corba and the OTS. The book ends with a chapter on Corba performance issues.

The chapters containing product descriptions will become out of date in a few years—the interfaces, that is, not the fundamentals. The author's approach, to design a case study and then apply it, is good, plus you can download the sources. This is a good way to extend the book's usefulness.

I recommend this book to developers and architects who need a transactional distributed system that uses Corba technology but don't have much experience in either. The author is surely an applied developer in the field and not a theorist. Nevertheless, theory is the starting point of every technology, and the author provides good, basic knowledge.

Angelo Bellotti is a software developer in the European Community Research Program on Distributed System Engineering. Contact him at Bellotti@sia-av.it.