



Components are "hot," and changing at a fast pace. Here are industry forecasts from two leaders in the field, followed by a state-of-the-practice overview and some of the latest work in CBSE.



Component-Based Software Engineering

Wojtek Kozaczynski, System Software Associates

Grady Booch, Rational Corporation

Pick up just about any software trade journal, and you'll find yourself inundated with something about components: there's Microsoft's DNA, Sun's Enterprise JavaBeans and Java Platform for the Enterprise, IBM's San Francisco project, the componentization of SAP's R/3—the list goes on. It's easy to see that a discipline is emerging, and yet there's little agreement on what "components" and "component-based software engineering" are. We thought it would be both interesting and useful to take a snapshot of this emerging domain so as to help *IEEE Software's* readers understand some of its key aspects.

In the last few years, components and component-based software engineering have gained substantial interest in the software community. Conceptually, CBSE unifies concepts from a number of software domains, such as object-oriented programming, megaprogramming, software architecture, and distributed computing. Indeed, it appears that the two main drivers for CBSE are the emergence of commercial distributed-computing infrastructures such as DCOM and Corba, and the increasing need for interoperability of independently developed software "chunks."

The rate of change in the software world is such that shortly after the ink dries on this issue, we can only hope to have captured a snapshot of a rapidly maturing domain. To this end, we were pleasantly surprised at the large number of

FURTHER READING

There are good books and Web sites that discuss different aspects of CBSE principles, technology, and processes.

Books

We recommend starting with *Component Software: Beyond Object-Oriented Programming* by Clemens Szyperski (Addison Wesley Longman, ISBN 0201178885). The book is slightly academic in its treatment of the subject, but provides a comprehensive review of existing component concepts and systems.

For a more practical description of component technology (in this case Microsoft technology), we recommend Roger Sessions' *COM and DCOM: Microsoft's Vision for Distributed Objects* (John Wiley & Sons, ISBN 047119381X). Sessions is an entertaining writer who makes difficult concepts accessible. He also provides an example of how to build a component system. For those looking for in-depth coverage of COM, see Don Box's *Essential COM* (Addison-Wesley Object Technology Series, ISBN 0201634465).

We also like *Building Business Objects* by Peter Eeles and Oliver Sims (John Wiley & Sons, ISBN 0471191760), which deals with business object and component analysis, design, and the construction process. Those who have read Sims' first book, *Business Objects* (McGraw Hill, ISBN 0077079574), should find this one particularly interesting.

We also recommend *Software Reuse: Architecture, Process and Organization for Business Success* by Ivar Jacobson, Martin Griss, and Patric Jonsson (Addison Wesley Longman, ISBN 0201924765). Although the title suggests that the book is not about components, the second section is an interesting overview of architectures of business component systems.

Of course the list would not be complete without referencing the popular John Wiley & Sons series by Robert Orfali and

colleagues. We recommend

- ◆ *Client/Server Programming with Java and CORBA, Second Edition* by Robert Orfali and Dan Harkey (ISBN 047124578X)
- ◆ *Instant CORBA* by Robert Orfali, Dan Harkey, and Jeri Edwards (ISBN 0471183334)
- ◆ *The Essential Distributed Objects Survival Guide* by Robert Orfali, Dan Harkey, Jeri Edwards, and Daniel Harkey (ISBN 0471129933)

WWW

For those who want to learn about Corba, we suggest Doug Schmidt's site (<http://siesta.cs.wustl.edu/~schmidt/>) rather than the OMG site (<http://www.omg.org>). Schmidt is a professor at Washington University's School of Engineering and Applied Science and a source of all kinds of interesting information about distributed computing. Once at the site, click on the Corba connection and just surf.

The key commercial players in the software component revolution also have good Web sites. Here are our favorites:

- ◆ the BEA site (http://www.beasys.com/m3/product_info/wp.htm). Look for a white paper on M3.
- ◆ the Microsoft DNA site (<http://www.microsoft.com/dna/>). Take a look at the technical overview of the DNA.
- ◆ the Microsoft COM and DCOM site (<http://www.microsoft.com/com/>). Proceed to the White Papers section.
- ◆ the IBM San Francisco project site (<http://www.ibm.com/Java/Sanfrancisco/>).
- ◆ the IBM Component Broker site (<http://www.software.ibm.com/ad/cb/>).
- ◆ the Sun Enterprise JavaBeans site (<http://java.sun.com/products/ejb/index.html>). Follow the documentation link.

We also encourage you to look up the references provided by the authors of the articles in this issue.

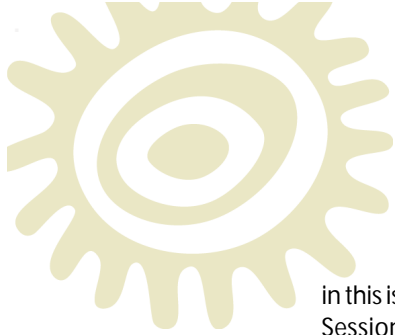
submissions we received from so many sectors of industry and academia. This is yet another indication that something interesting is going on.

WHAT IS A COMPONENT, ANYWAY?

That's a really good question. A definitive answer is hard to come by, largely because the different vendors of various commercial distributed-computing infrastructures take subtly different positions. Microsoft (with DCOM) and Sun (with Enterprise JavaBeans) lead the charge; there are material differences in the way these two component models address the semantics of interfaces, communication among components, threading, and distribution. The Object Management Group (with Corba) appears to be following, not leading, with an active request for proposal (orbos/97-06-12 CORBA Component Model RFP, Final Version) on a component model.

There's also the object-oriented versus component-based controversy, which further complicates the discussion. A few years ago a cover story in *Byte* dramatically announced that, basically, all things object-oriented were dead and components would soon rule. We don't take quite so harsh a position: clearly some differences exist, but there's also a tremendous amount of intellectual overlap between these two worlds.

As we were gearing up to write a comprehensive survey of the current state of CBSE, we were invited to the Workshop on Component-Based Software Engineering, which was held in conjunction with the International Conference on Software Engineering '98. The workshop brought together an excellent group of practitioners and academics to discuss this very issue: What is CBSE? Alan Brown and Kurt Wallnau, workshop organizers, wrote an excellent summary of the discussion. Since we are both strong proponents of reuse, we've included their summary



in this issue. In separate boxes in the overview, Roger Sessions takes a quick look at the leading infrastructure technologies, and David Chappell addresses how these approaches handle transactions and component state. The rest of the articles we've chosen to include are summarized on page 4.

THE FUTURE OF CBSE

It's always a bit risky to predict anything, the future of software in particular. It's easy to be blindsided by new technology that seemingly comes out of nowhere and suddenly gains critical mass. Just a few years ago Java was something you drank in the morning, and the web was something you cleaned out of the dusty corners of your house.

However, we can discern something about how CBSE came to be and gained prominence. Mega-programming, software architectures, and to a large degree object orientation came from the academic community and its small-environment, language-specific experiments. CBSE, on the other hand, has been driven from day one by everyday, real software problems. Software engineers voted with their keyboards in favor of OLE and Visual Basic, which were both received less than enthusiastically by purists but were in reality the first industrial implementations of CBSE concepts. Companies joined OMG by the hundreds to contribute to defining OO standards. Later on, software industry heavyweights joined forces in the so-called SONI group (Sun + Oracle + Netscape + IBM) to invest real and big money in promoting Java EJB.

What can we say about its future? It is rather predictable: nobody will be talking about it much in a few years—not because it will fail and go to the wayside but, to the contrary, because it will be mainstream software practice. To a degree it already is.

Moreover, CBSE will have a significant impact on our industry and hopefully on how we train software engineers. CBSE brings architecture thinking to the center of the software development process. It cleanly separates infrastructure from system logic and helps to cope with system complexity. It helps organize large-scale development and, most importantly, makes system building less expensive. So CBSE is here to stay.

Having said that, we offer the following few shorter-term observations. First, there will be continued entrenchment of COM and EJB, and developers will have to choose between them. Microsoft's

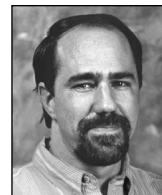
integration of COM+ and DNA and Sun's Jini will likely further that trend. Second, we'll see further componentization of software products, likely to be led by the various ERP (enterprise resource planning) vendors around the world as they work to improve the economics of their products. Third, component technology will begin to impact the process of software development. As teams gain more experience in building component-based systems, that experience will get translated into best practices for the software development process. And finally, we will see the biggest impact of CBSE shortly after the year 2000, when CIOs start getting rid of the legacy software that they learned to hate so much when modifying it to get through 1 January of the next millennium. ❖

About the Authors



Wojtek (Voytek) Kozaczynski is director of advanced development at the System Software Associates R&D Labs. SSA builds software solutions for manufacturing-sector companies. He is responsible for development of the next generation of BPCS, the SSA's family of ERP products encompassing basic manufacturing, supply chain management, and financial business processes. He is working on BPCS's runtime architecture, component development methods and tools, and the development of reusable software components.

Prior to joining SSA, Kozaczynski directed the Software Engineering Laboratory at Andersen Consulting's Center for Strategic Technology Research. In 1995–1996, he led a NIST-sponsored Advanced Technology Program research project on CBSE. He is a member of the Industrial Advisory Board of *IEEE Software*.



Grady Booch has been with Rational Software Corporation as chief scientist since shortly after its founding in 1980. He is coauthor of the Unified Modeling Language (UML), and his current work focuses on architectural patterns of complex systems. Booch is the author of six books, including *Object-Oriented*

Analysis and Design and *Object Solutions: Managing the Object-Oriented Project*.

Booch received a BS in computer science from the United States Air Force Academy and an MSEE in computer engineering from the University of California at Santa Barbara. He is a member of the American Association for the Advancement of Science, IEEE, ACM, Computer Professionals for Social Responsibility, the Airlie Software Council, and the DNRC. He is also an ACM Fellow and sits on the board of the Worldwide Institute of Software Architects.

Kozaczynski can be reached at wkozaczy@ssax.com. Booch can be reached at egb@rational.com.