Our 2005 Reviewers

The articles in *IEEE Software* are the result of many people’s hard work. We deeply appreciate the efforts of everyone who participated in our peer review process from 17 Nov. 2004 to 31 Dec. 2005. Authors often tell us how much they value reviewers’ comments and suggestions. Their expertise, care, and attention help maintain *Software’s* quality. If you’d like to contribute to our community by reviewing, visit www.computer.org/software/reviewers.htm for more information.

—Warren Harrison, Editor in Chief

Aynur Abdurazik, George Mason Univ.
Mark Aberdour, Open Source Testing
Silvia Abrahão, Univ. Politecnica de Valencia
Michael Adams
Art Akerman, Capital One
Jonathan Aldrich, Carnegie Mellon Univ.
Ian Alexander, Scenario Plus
Roger Alexander, Washington State Univ.
Deepak Alur, Sun Microsystems
Scott Ambler, Ronin Int’l
Paul Ambrose, Univ. of Wisconsin-Whitewater
Pierre America, Philips Research Lab
David Adams, Univ. of Arkansas
Mikio Aoyama, Nanzan Univ.
Joao Araujo, New Univ. of Lisbon
Aybuke Aurum, CAESAR
Paris Averiou, Univ. of Groningen
Jongmoon Baik, Information and Comm. Univ.
Arnaud Bailly, ETHZ
Elisa Baniassad, Chinese Univ. of Hong Kong
Sarah Beecham, Open Univ.
Klaas Berg, Univ. of Twente
Lodewijk Bergmans, Univ. of Twente
Tomas Berling, IT Univ. of Göteborg
Gabor Bertalan, Triad Computer Services
Danilo Beuche, pure-systems GmbH

Jean Bezivin, Univ. of Nantes
Konstantin Beznosov, Univ. of British Columbia
Robert Birdie, Carleton Univ.
Paul Black, Nat’l Inst. of Standards and Tech.
Maarten Boasson, Universiteit van Amsterdam
Conrad Bock, Nat’l Inst. of Standards and Tech.
Ron Bodkin, New Aspects of Software
Gary Boetticher, Univ. of Houston-Clear Lake
Terry B. Bollinger, MITRE
Egor Bondarev, Tech. Univ. of Eindhoven
Grady Booch, IBM
Jan Bosch, Univ. of Groningen
Mikhail Brikman, Salem State College
Giacomo Bucci, Università di Firenze
Shawn Butler, Carnegie Mellon Univ.
Zulema Caldwell, General Dynamics
Luiz Capretz, Univ. of Western Ontario
Collin Carbno, Saskatchewan Telecom
John M. Carroll, Virginia Tech
Lynn Carter, Carnegie Mellon West
Jeff Carver, Mississippi State Univ.
Richard Carver, George Mason Univ.
Ana Regina Cavalcanti da Rocha, Federal Univ. of Rio de Janeiro
Alejandra Cechich, Univ. of Castilla-La Mancha
Marcos Chaim, Univ. of São Paulo
Michel Chaudron, Tech. Univ. of Eindhoven
Shigeru Chiba, Inst. of Information Science and Electronics
Evan Chowdhury, Univ. of Maine
Sunita Chulani, Center for Software Eng.
Lawrence Chung, Univ. of Texas at Dallas
Bradford Clark, Software Metrics
Paul Clements, Carnegie Mellon Univ.
Yvonne Coady, Univ. of Victoria
Robert Cochran, Catalyst Software
Tal Cohen, Technion
Annie Combelles, Q-Labs
James Cooper, IBM T.J. Watson Research Center
Vittorio Cortellessa, Università dell’Aquila
Ivica Crnkovic, Maldalen Univ.
Bojan Cukic, West Virginia Univ.
Y.S. Dai, Purdue Univ. School of Science

Maya Daneva, Univ. of Twente
John Dean, Dean Consulting Services
Geert Deconinck, Katholieke Univ. Leuven
Fons de Lange, Philips Research
Onur Demirors, Middle East Technical Univ.
Daniel Deveaux, Université de Bretagne-Sud
Kris De Volder, Univ. of British Columbia
Maja D’Hondt, Vrije Universiteit Brussel
Theo D’Hondt, Vrije Universiteit Brussel
Oscar Dieste, Univ. Complutense de Madrid
Adair Dingle, Seattle Univ.
Torgeir Dingsoyr, SINTEF
Paolo Donzelli, Univ. of Maryland
Stefan Drees, Drees Consulting and Lecturing
Geoff Dromey, Software Quality Inst.
Yucong Duan, Chinese Academy of Sciences
Yael Dubinsky, Technion
Eric Dubois, Centre de Recherche Public Henri Tudor
Tore Dyba, SINTEF
Fedor Dzerjinski, Rossiiyskiy Kredit Bank
Anthony Earl, Sun Microsystems
Steve Easterbrook, Univ. of Toronto
Christof Ebert, Alcatel HQ
Alexander Egyed, Teknowledge
Santiago Elbe, Univ. Politécnica de Madrid
Nancy Eickelmann, Motorola
Eric Eide, Univ. of Utah
Khalel El-Emam, Inst. for Information Technology
Islam El-Maddah, Ain Shams Univ.
Tzilla Elrad, Illinois Inst. of Technology
Robert Erbacher, Utah State Univ.
Fernando Fernandez, Univ. Carlos III Madrid
Luis Ferreira, Politecnica Inst. of Porto
Robert Filman, NASA Ames Research Center
Francesco Flammini, Univ. Federico II of Naples
Xavier Franch, Universitat Politecnica de Catalunya
Bernd Freisleben, Univ. of Marburg
Lidia Fuentes, Universidad de Malaga
Owen Funkhouser, Dynetics
Tsuneo Furuyama, Tokai Univ.
Alexander Fyukov, Philips Research Lab
Rose Gamble, Univ. of Tulsa
With a suitable behavioral representation to detect requirements problems early, this behavior can provide the ladder to climb over the brick wall—scaling the wall results in the overall system exhibiting integrated behavior. What Brooks calls the “essence” of software engineering, Brooks suggests it’s unlikely that there will be any “inventions that will do for software productivity, reliability, and simplicity what electronics, transistors, and large-scale integration did for computer hardware.” In other words, “building software will always be hard. There is inherently no silver bullet”—we’ve run into a brick wall.

Scaling the wall

Faced with a situation like this, our greatest challenge in advancing any discipline is always to break free from the shackles of our past. In this regard, David Harel’s advice provides a signpost to where software engineering is and should be heading: “It is our duty to forge ahead to turn systems modeling into a predominantly visual and graphical process.”

What Brooks calls the “essence” of software entities has little to do with the conceptual view of systems. Systems are built out of a network of interacting components (some of which might be systems in their own right). Such a view implies all systems might have designs that can be embedded in space. It doesn’t matter whether we’re talking about systems we intend to implement in software, hardware systems, other physical systems, business systems, or any other conceptual systems. In all cases, the system components encapsulate and exhibit individual behavior, and they interact by passing control and data to other components. This results in the overall system exhibiting integrated behavior.

An appropriate representation of this behavior can provide the ladder that lets us climb over the brick wall—to get complexity and change under control, to overcome the so-called invisibility of software, to make gains with conformity, and, as a side benefit, to detect requirements problems early. With a suitable behavioral representation, we can systematize and simplify...