Offsetting Human Limits
With Debugging Technology

By Thomas G. Moher and Paul R. Wilson,
University of Illinois at Chicago
OFFSETTING HUMAN LIMITS WITH DEBUGGING TECHNOLOGY

Debugging is about human frailty. Our natural limitations in reasoning power, memory, and communication are both the reasons we debug and why the activity of debugging itself can be so difficult. Viewed as artifacts of our programming culture, debugging tools reflect the strategies we have found to be effective in locating and correcting program errors. Changes in language and system technologies in turn create the need for new problem-solving strategies, and the cycle continues.

The debuggers described in this issue are not intelligent tools, at least with respect to fault localization or remediation. Instead, they are visibility tools, making available information that is usually lost or hidden from programmers. These tools let programmers craft their views to reduce potentially vast information spaces to manageable (and meaningful) proportions.

Each article describes experience with practical debugging tools. We found it worthwhile to pay particular attention to the debugging examples to get the real flavor of the power and limitations of the tools presented here.

Takao Shimomura and Sadahiro Isoda report on recent extensions to their VIPS debugger, originally described in *IEEE Software* in May 1987. In the original version of VIPS, visualization of linked structures was limited to very small lists or to selected portions of larger structures. The newer version of VIPS extends the original work by adding multiple levels of browsers, letting users interactively identify sublists of interest. Animation facilities support the visualization of dynamic list operations simultaneously among the various views.

Hiralal Agrawal, Richard DeMillo, and Eugene Spafford present a system for selective checkpointing of computational sequences, letting users backtrack from checkpoints without the need to reexecute the program to reach recent prior states. In contrast to more comprehensive (and storage-intensive) checkpointing schemes, they constrain backtracking to limit storage requirements. The resulting debugger offers a structured view of dynamic events, similar to lexical scope rules’ effect on static visibility.

Keijiro Araki, Zengo Furukawa, and Jingde Cheng argue for prescriptive, systematic models of the debugging process. Their work echoes research in cognitive aspects of programming, in that debugging is characterized as an iterative process of synthesizing, testing, and refining hypotheses about bug locations and repairs. The authors argue that debugging tools should provide direct support for each of these components of the process, and they describe how a debugging tool for Ada programs seeks to meet these criteria.

David Rosenblum describes the Task Sequencing Language, which lets programmers specify constraints on the behavior of concurrent programs. The constraints, specified by high-level annotations to Ada programs, are monitored at runtimes; when a violation is detected, control is returned to the user, along with information about the nature of the violation. Rosenblum argues explicitly that such a tool is a necessary crutch for human reasoning capabilities when dealing with the complexities of concurrent process interaction.

What is the future of debugging?

Not surprisingly, the outlook is bullish. While work in formal development methods and tools continues with vigor, the industrial and commercial worlds remain largely dependent on development methodologies in which debugging is a necessary component.

Someday, perhaps, most debugging will take place in the specification domain rather than the code domain, with the emphasis shifting to declarative rather than procedural fault localization and repair. Be...
that as it may, people will continue to make mistakes, and the products of human ingenuity will never be free of the need for corrective mechanisms. The principles of visibility demonstrated by the articles in this issue — multiple views at varying levels of abstraction, direct support for backward-chaining problem solving, isolation of concurrent threads of execution, and the like — will remain important as long as people are involved in the development process.

Thomas G. Moher is an associate professor of computer science at the University of Illinois at Chicago. His research interests include human factors in software development, programming environments, and interactive systems. Moher received a BS in computer science from the University of Michigan and a PhD in computer science from the University of Minnesota. He is a member of the ACM, IEEE Computer Society, and Human Factors Society.

Paul R. Wilson is completing his PhD dissertation in computer science at the University of Illinois at Chicago. His research interests include languages and environments and their relationships to operating systems and computer architectures. Wilson received a BS in computer science and a BS in psychology from Regents College of the State University of New York and an MS in electrical engineering and computer science from the University of Illinois at Chicago. He is a member of the ACM, IEEE, and Cognitive Science Society.

For turning data into information, two books by Edward Tufte on the state of the art for graphical displays

**ENVISIONING INFORMATION**

"A remarkable range of examples for visual thinking. A real treat for all who reason and learn by means of images." RUDOLPH ARNHEIM

"A savvy guide for the professional. Beautifully illustrated, well-argued.” SCIENTIFIC AMERICAN

"A wonderful book.” COMPUTER GRAPHICS AND APPLICATIONS

Over 400 illustrations with exquisite 6- to 12-color printing throughout. Finest examples in technical, creative, and scientific presentations: diagrams, legal exhibits, computer graphics, charts, maps, illustrations, use of color. Presenting complex material clearly. $48 per copy postpaid.

**THE VISUAL DISPLAY OF QUANTITATIVE INFORMATION**

This map portrays the losses suffered by Napoleon's army in the Russian campaign of 1812. Beginning at the left on the Polish-Russian border near the Niemen, the thick band shows the size of the army (422,000 men) as it invaded Russia. The width of the band indicates the size of the army at each position. In September, the army reached Moscow with 100,000 men. The path of Napoleon's retreat from Moscow in the bitterly cold winter is depicted by the dark lower band, which is tied to a temperature/time scale. The remains of the Grande Armée struggled out of Russia with only 10,000 men. Six dimensions of data are displayed on the two-dimensional surface of the paper. (Two-color poster, $12 postpaid.)

"The century's best book on statistical graphics” COMPUTING REVIEWS

"A truly splendid volume...so much care in its writing, illustration, typography, and production. It is among the best books you will ever see.” DATAMATION

250 illustrations of the best (and a few of the worst) statistical charts, graphics, tables, with a detailed analysis of how to display quantitative data for precise, quick, effective analysis. Highest quality book design and production. $36 per copy postpaid.

Moneyback guarantee. Order directly from publisher, enclosing check: Graphics Press Box 430-S Cheshire, Connecticut 06410