tributed workstation environment. In P. Henderson, editor, *SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, pages 104–112, Pittsburgh PA, April 1984. Special issue of *SIGPLAN Notices*, 19(5), May 1984.

[2] H. F. Korth and A. Silberschatz. *Database System Concepts*. McGraw-Hill, New York, 1986.

[3] M. Stonebraker, E. N. Hanson, and S. Potamianos. The POSTGRES rule manager. *IEEE Transactions on Software Engineering*, 14(7):897–907, July 1989.

[4] D. Garlan. Extending IDL to support concurrent views. *SIGPLAN Notices*, 22(11):95–110, November 1987.

[5] R. Snodgrass and K. Shannon. Supporting flexible and efficient tool integration. In R. Conradi, T. M. Didriksen, and D. H. Wanvik, editors, *Advanced Programming Environments*, volume 244 of *Lecture Notes in Computer Science*, pages 290–313. Springer-Verlag, Trondheim, Norway, 1986.

[6] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, 12(2):213–253, June 1980.

[7] W. Harrison. $RPDE^3$: A framework for integrating tool fragments. *IEEE Software*, pages 46–56, November 1987.

[8] D. Yellin and R. Strom. INC: A language for incremental computations. In *SIGPLAN '88 Conference on Programming Language Design and Implementation*, pages 115–124, Atlanta GA, June 1988. Special issue of *SIGPLAN Notices*, 23(7), July 1988.

[9] A. N. Habermann, L. Flon, and L. Cooprider. Modularization and hierarchy in a family of operating systems. *Communications of the ACM*, 19(5):226–272, May 1976.

Box E: Other Systems

The paradigms considered above are but a brief sampling of the kinds of systems based on tool abstraction. Other examples include monitors in the Domain Software Engineering Environment [1], views and rules in relational database systems [2, 3], some extensions to the Interface Description Language [4, 5], and AI blackboard architectures [6].

One of the major goals of the RPDE^3 software development environment is "to support both the integration of tools constructed ¿from many small fragments and the construction of tools that can be extended to process new types of data without source-code changes" [7]. The most significant limitation of RPDE^3 with respect to tool abstraction is its lack of event-driven tool invocation. Tool interleaving is instead carried out using explicit message passing and routing rules. The INC programming language has been developed for writing incremental computations [8]. INC is based on the observation that "there are many problem domains that require repeating a computation many times, each time on slightly different data", and the claim that the specialized techniques for efficient recomputation developed for such domains can be unified in a language-based solution. INC program components are thus tiny toolies, with event-driven integration specified by a circuit graph of components.

Even some systems that do not support tool abstraction take a similar approach with regard to opening up the representation of data to a variety of computational entities. In the FAMOS family of operating systems, for instance, different levels of virtual machines can share representations of abstract data types by having a module extension "join" a module to get access to its internal representation [9].

References

[1] D. B. Leblang and R. P. Chase, Jr. Computer-aided software engineering in a dis-