Turning points in interaction with computers

by F. E. Allen

Humans and computers have interacted since the early days of computing through some type of interface. As computing has evolved, so have the interfaces. In 1981 Lewis Branscomb stated in "The Human Side of Computers," the foreword to the *IBM Systems Journal* special issue on human factors,[1] that "So long as computers were simple and operated only by specialists, the human interface was relatively unimportant. The frustrations caused by complex procedures and software idiosyncrasies were borne by experts motivated to accept the challenge. But all that has changed. . . . As a result, information systems designers are increasingly being called upon to develop products that are easier to install, easier to service, and—above all—easier to use." Later on in the same discussion, Branscomb says: " . . . the real bottleneck in access to computing is not hardware but ease of programming. Ideally, users should not feel they are 'writing' a program at all when they ask for information from a computer system." And he continues at the end of the foreword: "The challenge to our industry for this decade is to bring computer capability—usefully and simply—to people without special training. If we succeed, future generations will acknowledge that the information systems born in this century quickly became the most supportive tools, compatible with human needs, that mankind has ever devised."

Discovering essays like Branscomb's was one of the many delightful rewards of looking in the *Systems Journal* for papers representing turning points in interaction with computers. His essay captures the hopes and challenges expressed one way or another in many of the papers related to this very broad topic. The evolution of the area over time is fascinating.

Early papers struggled to make the subject scientific and respectable by focusing on foundational issues and methods; later papers are more comfortable reporting experiences and making observations. The kinds of interactions were initially very limited (e.g., keyboards and primitive displays), then grew to encompass many different forms, from programming languages to graphics and multidimensional visual interfaces. Now the interactions are increasingly multi-modal, with speech, image, sound, text, and touch being just some of the interfaces being seamlessly integrated into user-friendly interfaces. Are these tools "compatible with human needs"? I leave that to the reader to decide. It is clear, however, that some of the early papers were surprisingly prescient.

The interaction turning points identified in this section of this retrospective issue are obviously not inclusive, nor are the topics considered comprehensive. The three topics chosen here have made a difference, and, in most cases, their real value has yet to be realized. In the first topic, programming languages, the power of modeling and simulation via executable models is seen as an old idea whose time is yet to come. The second topic, automating the office, was first discussed in the *Systems Journal* in 1979 and is still under development, albeit in much larger contexts: forms processing across networks and multiple enterprises doing supply chain

management. Though the third topic, human factors, was considered among the early papers, today we are still trying to quantify and predict ease of use.

Programming languages

Programming languages have been the *lingua franca* of computer interaction since the early 1950s. Starting with the use of symbols to represent machine codes and data addresses, far more sophisticated languages evolved rapidly. By 1960, LISP, COBOL, and FORTRAN, to name just a few, had strong advocates and many users. They differed greatly in syntax (how programs were written), semantics (the meaning of what was written), and programming models. Ideally the syntax is natural to the solution domain, the semantics enables precise expression of the solution, and the programming model matches the way a user thinks about a problem. However, many other factors influence language designs. The FORTRAN and COBOL languages for scientific and commercial users, respectively, are examples of early languages whose characteristics differed significantly but were hugely successful in their own problem domains.

Of the multiplicity of interesting early languages, two others, APL and SIMSCRIPT, also represent turning points in computer interaction and were the subject of papers in the *Systems Journal*. In "Programming Notation in Systems Design," reprinted in this section, Ken Iverson, the creator of APL, observes that precise and formal modeling of complex systems, i.e., systems composed of elements from disparate disciplines, would benefit from a specification of the procedures or algorithms executed by the system when this is possible. The paper then shows that APL provides an excellent model specification language for complex systems since, among other things, it is precise, formal, and executable. The important turning point observation here is that the same language can be used to both precisely model a complex system and simulate its execution. Having mastered the syntax and semantics of APL, the user could succinctly specify the components of an existing or proposed system, then execute the model to evaluate behaviors. The SIMSCRIPT language, having a very different ancestry stemming from economics rather than mathematics, was developed in the same time period by Harry Markowitz (a subsequent Nobel laureate). The paper, "A Description of the SIMSCRIPT Language," by Bernard Dimsdale and Markowitz, reprinted here, discusses the interplay of modeling and simulation in understanding complex systems such as managing supermarket checkout queues to optimize profits. Again the ability to use the same language to both model the system and simulate execution scenarios is identified as important.

In "NIL: A High-Level Language for Distributed Systems Programming," included in this section, Francis Parr and Robert Strom describe a language for defining executable architectures. Their Network Implementation Language system enabled network protocol specification with guaranteed error isolation and the enforcement of interfaces. NIL was another interesting step in the continuing effort to empower experts with the interfaces and tools needed to solve problems. Unfortunately the modeling and simulation languages have not developed as rapidly as needed. However, the recent emergence of powerful computer and communications capabilities, together with improved models for many complex systems ranging from bomb designs to proteins to semiconductors, has created widespread interest in using these capabilities to build computer models and simulators of even more complex systems. The pioneering work represented in these papers definitely contributes to the establishment of executable models of complex systems, enhancing our ability to understand, predict, and reason about their behaviors.

Automating the office

Automating manual activities was the motivation for building the first computers and a continuing activity ever since. In the commercial environment, automating routine business processes such as personnel and payroll management was one of the early computerized business processes. The challenge of automating the processes associated with an office involved automating a very different set of activities, including the timely and selective creation and circulation of documents so that appropriate actions could be taken. A paper by Gruhn and Hohl[2] describes the components of such an activity. A business language called Office-by-Example is the subject of a subsequent paper by Zloof.[3] He describes OBE as requiring little training to use yet being powerful enough to express most office activities. It used a nonprocedural, two-dimensional in-

terface to mimic the manual procedures of business and office systems. A later paper, "Visual Programming: Perspectives and Approaches" by Nan Shu, included here as a reprint, lays the conceptual background for assessing and understanding visual languages. These papers, published between 1979 and 1989, go from identifying the components of an interesting domain—office automation—to the beginnings of foundational work characterizing a general form of user-friendly interface languages applicable to a variety of domains. Efforts and interest in automating manual activities now go well beyond anything imagined even a few years ago. The convergence of computing, communications, and digitized information has enabled the integration of processes distributed globally. Supply chain management is an example of such an integrated process often distributed across multiple business units. Ways of developing and interacting with these complex systems remains a challenge.

Human factors

The human factors discipline studies what constitutes the best interfaces between people and machines. Emerging as an experimental science during World War II with a focus on ergonomics, the understanding of the physical factors of using such things as displays and keyboards became quite advanced early on. However, quantitative methods were slow to emerge, leading to difficulty in building predictably good interfaces for the intended use.

IBM's commitment to better human-computer interfaces (HCIs) and to assessing their effectiveness in human factors studies is the subject of numerous papers in the *IBM Systems Journal*. In an early paper by Stephen Boies titled "User Behavior on an Interactive Computer System," included in this section, the author uses methodologies from behavioral sciences to understand the human performance on a complex interactive computer system: the System/360* Time Sharing System. The definitive results provided useful input to designers of the system and, more importantly, the scientific approach used helped establish human factors as a valuable tool for assessing and predicting usability. An even earlier effort in IBM was the Human Factors Center at IBM's development laboratory in San Jose, California. It was established to "provide management with objective and comprehensive evaluations of products or systems to ensure that man-machine or man-system interfaces have been optimized and that any human factors risks have been assessed and minimized."[4] The 1981 paper by Hirsch[4] is a wonderful compendium of scientific methodologies, empirical results, observations, and recommendations resulting from 20 years of experience studying the human factors associated with keyboards, including those for Japanese and for blind people, displays, programming languages, text editors, and on-screen windows.

Though interactive modes and contexts have changed significantly since these papers were published, the overall approach to the study of human factors is a continuum of the experimental, multidisciplinary methodologies reported in these two papers. "Every-citizen" interfaces have become a Holy Grail sought after by solution and service providers everywhere. They are often the critical differentiators in acceptance by intended users whether they are first-graders or rocket scientists. Predictable, useful interfaces also remain a Holy Grail, and methods for achieving this goal are continuing to use and expand the approaches advocated in the papers cited.

Concluding remarks

The development of interfaces by which humans and computers interact has brought about some of the most significant turning points in computing in the last 38 years. The work on interfaces in programming languages, office automation, and human factors has enabled more and more people to interact effectively with computers, but there is more work to do. The *IBM Systems Journal* has published many papers that reflect results of work in this area, and readers can see a representative selection of those papers in this section of this issue.

*Trademark or registered trademark of International Business Machines Corporation.

Cited references

1. L. M. Branscomb, "The Human Side of Computers," Foreword, *IBM Systems Journal* **20**, No. 2, 120–121 (1981).
2. A. M. Gruhn and A. C. Hohl, "A Research Perspective on Computer-Assisted Office Work," *IBM Systems Journal* **18**, No. 3, 432–456 (1979).
3. M. M. Zloof, "Office-by-Example: A Business Language That Unifies Data and Word Processing and Electronic Mail," *IBM Systems Journal* **21**, No. 3, 272–304 (1982).
4. R. S. Hirsch, "Procedures of the Human Factors Center at San Jose," *IBM Systems Journal* **20**, No. 2, 123–171 (1981).

Frances E. Allen *IBM Research Division, Thomas J. Watson Research Center, P. O. Box 704, Yorktown Heights, New York 10598 (electronic mail: franalle@us.ibm.com).*

Ms. Allen is an IBM Fellow at the T. J. Watson Research Center where she is the senior technical assistant to the Research Division Vice President of Services and Solutions. She was the 1995 president of the IBM Academy of Technology and specializes in compilers, compiler optimization, and high-performance systems. Her work includes algorithms and technologies that are the basis for the theory of program optimization and are widely used throughout the industry. Ms. Allen is a member of the National Academy of Engineering, and a Fellow of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, and the American Academy of Arts and Sciences. She is on several outside boards including the Computer Science and Telecommunications Board of the National Research Council and the CISE (Directorate for Computer and Information Science and Engineering) Advisory Board of the National Science Foundation. She holds an M.A. degree in mathematics from the University of Michigan and an honorary D.Sc. degree from the University of Alberta.

Reprint Order No. G321-5701.