Anecdotes

James E. Tomayko, Editor

[*Editor's Note: There are recent initiatives to pursue more aggressively the history of software. For instance, the Charles Babbage Institute and the Heinz Nixdorf Museum cosponsored a conference on this subject in Paderborn, Germany, in April 2000. A report about this conference will undoubtedly appear soon in Events and Sightings. In that spirit, I asked Michael Jackson to give us his memories of the origins of the popular software development methods most closely associated with his name.*]

The Origins of JSP and JSD: A Personal Recollection

Michael Jackson

I performed my first serious programming work in the very early 1960s in Assembler languages on IBM and Honeywell machines. Although I was a careful designer—drawing meticulous flowcharts before coding—and a conscientious tester, I realized that program design was hard and the results likely to be erroneous. I wrote some assertions in the Honeywell programs that formed a little system for an extremely complex payroll, with runtime tests that halted program execution during production runs. Time constraints did not allow restarting a run from the beginning of the tape. So for the first few weeks, I had the frightening task on several payroll runs of repairing an erroneous program at the operator's keyboard—correcting an error in the suspended program text, adjusting the local state of the program, and sometimes modifying the current and previous tape records—before resuming execution. On the Honeywell 400, I could do all this directly from the console typewriter. After several weeks without halts, there seemed to be no more errors. Before leaving the organization, I replaced the runtime halts with brief diagnostic messages: not because I was sure all the errors had been found, but simply because there would be no one to handle a halt if one occurred. An uncorrected error might be repaired by clerical adjustments; a halt in a production run would certainly be disastrous.

In 1964, I went to work at John Hoskyns and Company, a new consulting firm in London. Seeking a more reliable and systematic way of programming, we spent a lot of time thinking about the problems of program design. In those days, disk drives were quite new and most often were used to hold sequential files (like the tape files that were the standard basis of data processing systems). So program design seemed to be chiefly about sequential processes. Barry Dwyer joined the company in 1966, and Brian Boulter joined soon after. We worked together on improving our design methods. We experimented with top-down decomposition and with some notions rather similar to coupling and cohesion. However, these notions did not seem to hold the key to design. Eventually we saw, in somewhat vague terms, that the coding discipline of structured programming invited the adoption of a hierarchical program structure to match the structures of the data files. Sometimes those data structures were incompatible, but we were not sure how to handle this difficulty. We also recognized (it was Dwyer's idea) that backtracking was an important technique that everyone else (practitioner or academic, working on program design methodology) seemingly ignored.

A vital stimulus to our work on program design was a project—the Microsystem—we did for a large insurance broker. The application demanded daily processing of small numbers of many different transaction types, each with its own pattern of access to the many master files. An online transaction-based solution was not economically practical at the time; the low volumes and varied access patterns made it impossible to design an efficient batch system. The Microsystem worked as a dynamically scheduled batch system. Transactions were run against one file until they needed access to another. They were then suspended and sorted into the appropriate order for the next run. Alternating runs and sorts continued until all the transactions in the batch were completed. In this way, the execution of the system was scheduled according to the needs of the current batch of transactions, rather than in a fixed pattern of system flow. Dwyer did most of the detailed design of Microsystem, and many of the JSP ideas had their origin there.

Around this time, in the mid 1960s, larger random access memories (as large as 256 Kbytes) were becoming commonplace, and the size of data processing programs increased dramatically. As programs grew larger, monolithic program texts became less and less practical. Modular Programming became a widely recognized concern, and the first shoots of the Structured Revolution began to appear. Larry Constantine ran a Modular Programming Symposium in 1968, with contributions from himself and from George Mealy, along with several others. Dwyer and I found an audience in the United Kingdom and the United States for seminars on our program design ideas.

At the end of 1970, I left Hoskyns and started my own

 1058-6180/00/$10.00 © 2000 IEEE

[JSP-Cobol] allowed a program to be designed as a tree of sequential processes, communicating by writing and reading virtual sequential data streams.

company, Michael Jackson Systems Limited, working alone. In 1971, I was invited to give a program design seminar at a Wall Street bank, a series of two-day Professional Development Seminars in the United States for the Association for Computing Machinery, and then a series of one-week courses in the United Kingdom, partly sponsored by the British Computing Society. In developing the material for these courses, and in argument and discussion with the course participants, the design method later known as JSP was worked out in detail. There was a more exact diagrammatic notation for the regular structures and a more exact correspondence between data and program. Incompatibilities between data structures, now classified in three kinds of structure clash, were more surely identified and resolved. The JSP coding technique of program inversion—something like an implementation of the semicoroutines of Simula 67 that was practical for Cobol programs—was worked out in detail. The correct placing of operations, especially read operations, in the program structure was systematized. The backtracking technique was embodied in three design steps: first ignoring the need for backtracking, then introducing the necessary GOTO (the "quit statement"), and finally dealing carefully with the side effects.

It all worked very well for sequential programs. The company began to grow. Starting with Sweden, we licensed our course and method material in many European countries, in the Americas, and in the Far East. In 1974, the UK government adopted JSP (under the name SDM) as its standard program design method. Boulter and I designed and built a preprocessor for Cobol that we called JSP-Cobol. It allowed a program to be designed as a tree of sequential processes, communicating by writing and reading virtual sequential data streams. Using program inversion, any process of the tree could be chosen as the root, and the whole tree mechanically generated as one or many Cobol compilation modules. In keeping with Cobol style, the preprocessor language provided an Implementation Section, in which arbitrary low-level implementations of the basic sequential write and read primitives could be separately specified for each real or virtual data stream, to allow compatibility with CICS, IMS, and other externally defined environments. Essentially, the method and its supporting tools abstracted the general notion of a sequential data stream from its many particular embodiments in tape and disk files, printer files, sequences of calls of a procedure, streams of messages in a communication channel, sequences of accesses to database records, and even sequences of low-level interrupts.

I described JSP—though not under that name—in the book *Principles of Program Design*, published in 1975.[1] The ideas of JSP began to reach a wider audience through this book and its translations—in German, Japanese, Dutch, Spanish, and Portuguese—and also through other authors' books on JSP. The method also attracted academic attention and was taught in a number of university courses.[2]

As the company grew, we began to extend JSP, developing it into the method for specifying and implementing information systems that was eventually named JSD. An information system could be seen as a simulation or model of the real world, with added functionality to provide the information outputs. JSD viewed the real world as a collection of entities such as customers, products, or accounts. Each entity has a long-term history of events, and this history forms a sequential process. Execution of the process can be resumed and suspended for each transaction or for each batch, applying to entity processes the technique the Microsystem had applied to transactions. Communication among entities is by shared events; the local variables of the processes are the "state vectors" of the entities. In the eventual implementation, the state vectors become the entity master records, the events form the transactions, and scheduling of the sequential processes becomes the responsibility of batch program shells. Essentially, this was an object-oriented, or object-based, view of the system: The JSD entities are objects, and the program texts executed for the transaction types implement the objects' methods.

In 1977, John Cameron joined the company, and we worked together on the development of JSD, refining the underlying ideas into a coher-

ent systematic procedure for system analysis, specification, and design. We gave the very first JSD courses in 1979 and early 1980. These were highly experimental presentations of a very incomplete method. Later we succeeded in filling in many of the gaps and repairing some of the defects, and public course presentations began in the summer of 1980. I described the method in the book *System Development*,[23] published in 1983. In the same year, Cameron wrote his IEEE Tutorial Text on JSP and JSD. The use of JSD spread from batch data processing systems to interactive and embedded systems. Specifically, it proved effective for simulation and command-and-control systems. For example, it was used very successfully in the simulation of a fly-by-wire helicopter and in the development of command-and-control software for a submarine system.

In spite of these successes, I had been gradually coming to recognize that JSP and JSD were less universal in their application than we had at first supposed. This was not a defect. It was an important technical strength: Effective software development methods must be sharply focused to exploit the characteristics of particular classes of problems and systems. After 1984, my personal interests shifted into broader software engineering concerns about problem and method classification and structure; principles of description in software development; and the underlying basis in reality for requirements, specifications, and other descriptions of the kinds that software developers produce and manipulate. Cameron and others continued to work on JSD and began to focus on a more explicitly object-oriented version of the method. The company developed some tools to support JSD, but these tools were less successful than the Cobol preprocessor and other JSP tools our Swedish licensees and the Atomic Energy Research Establishment at Harwell developed.

Since 1970, many people contributed to the development of JSP and JSD. Among them were Alan Birchenough, Tony Debling, Andrew Farncombe, Leif Ingevaldsson, Jacqueline Kathirasoo, Ashley McNeile, Alan Moore, Hans Naegeli, Dick Nelson (who introduced the name JSP), Jim Newport, Bo Sanden, Peter Savage, Ray Scott, Mike Slavin, and many others. Today, JSP and JSD are still in use in Europe and the United States, but in a relatively small number of organizations. New tools are still being built.

Michael Jackson
101 Hamilton Terrace
London NW8 9QY, England

Acknowledgments
Alan Birchenough reminded me of some of our successes. Larry Constantine gave me some information about the Modular Programming Symposium of 1968. Barry Dwyer gave me much help, reminding me of many things I had forgotten or only dimly remembered. Andrew Farncombe confirmed several points and corrected a mistake in my early history of JSD. Daniel Jackson helped me to clarify some obscurities and omissions in my account. Ray Scott confirmed my recollections of the late 1970s and the 1980s.

TRADIC

Frank S. Preston

[*Editor's Note: Frank Preston has more on the TRADIC computer described in a recent issue.*]

This relates to two articles on TRADIC in the *Annals* last year.[24,25]

In about 1954, Cdr. Lemos (Navy Bureau of Ordnance, later Bureau of Air) asked me to visit Bell Laboratories in Murray Hill, New Jersey, to see and appraise a digital computer for bombing. I was the project manager for the Norden Laboratories in charge of development of the AN/ASB-1 bombing system, just then becoming operational for Navy attack planes (AJ-1 and A-3D). This visit was for a single day.

I can place the date after May 1953, because I had bought a Curta Calculator in Switzerland that I needed to check results produced by our CP-66/ASB-1 computer, an analog bombing and navigation system. I carried the Curta in my attaché case in its waterproof container, which is about the size of a grenade. When the guard at the front desk looked in my case, he grabbed the calculator and started to run with what he thought was a bomb. With great difficulty, I persuaded him to stop running and let me open it. (He could not open it because of the left-handed thread on the case.) Because of other events, I believe this was before 1955.

It was explained to me that Western Electric promoted the trip because it had proposed a version of TRADIC to the Navy for some application—not necessarily for bombing. Despite that, I was asked to comment on the application to a bombing system. We were shown something smaller than the Leprechaun computer (Harris,[24] see Fig. 3 on page 53) that performed some navigation problem. This computer was smaller and much lighter than the size quoted in Brown's[25] Table 4 (on page 57) or Brown's[25] Fig. 6 (on page 59). We were told that it computed a complete bombing solution every second and that this computer

Westinghouse Electric proposed its own potentiometers at $10,000 each, whereas ours cost under $500.

was interpolated to give 100 results per second.

Our evaluation was that the computer showed great promise. Our primary concern was that Bell Labs had decided to do everything with transistors. This worked in the logic and arithmetic sections but made the storage and other portions of the computer less desirable than other technologies then available. Also, because of limitations on aircraft carriers, considerable improvement would be needed before an operational unit could be available. (Because of being carrier-based, the Norden system required special provisions in the plane and the system for boresighting and for interchanging components compared with the Air Force systems.) The Norden system was both radar and visual (with a periscope), and the elevation angle computer was half in the periscope and half in the computer to provide the angle transfer. We had had Helipot (Beckman) develop custom potentiometers (accurate to 0.025 percent) for this. (See *IRE Transactions,* vol. EC-4, no. 3, p. 101, Sept. 1955,[26] and patent 2,738,934.) Westinghouse Electric proposed its own potentiometers at $10,000 each, whereas ours cost under $500.

More significant as an illustration of this stage in development is that the military did very little to encourage technical exchanges between contractors, and the competitive situation made this visit a rarity. Brown[25] discusses in his introduction the Norden Optical Bomb Sight (Mark 15). Manufacture of this ceased in May 1945, although it was in operational use for a while later. When we started to develop postwar bombsights, we wrote the specifications for the Navy, and these specifications were given to the Army Air Corps at the time it was converted to the Air Force. This served as the start of the separate development of bombsights. The Navy and Air Force became engaged in a competition for the assignment to deliver tactical and strategic (atomic) bombs, so we learned little of Air Force bombing development. The Navy chose to use the same bombsight in various aircraft, whereas the Air Force went to a weapons system type of contract earlier than the Navy. The result was that the Norden systems benefited from the improved capability due to longer operational use, and the Air Force systems tracked new technology more closely. We introduced a digital computer (DDA) in about 1956 to replace the analog one. This was earlier than the announced Air Force digital bombing systems.

Frank S. Preston
88 Notch Hill Road, Apt. 324
North Branford, CT 06471

Lovelace–Babbage Letters Discovered in Newcastle

Christopher Goulding

The discovery of a hoard of unpublished letters in Newcastle upon Tyne, England, has shed new light on the work of 19th-century computer pioneer Charles Babbage and his relationship with Augusta "Ada" Byron Lovelace (1815–1852), daughter of poet Lord Byron.

The letters were unearthed in the Brooks Manuscript Collection, a large Victorian autograph collection owned by the Society of Antiquaries of Newcastle upon Tyne and held at Northumberland County Record Office, which I was looking through as part of my postgraduate English literature research. Among 12 leather-bound volumes of papers relating to hundreds of literary and historical figures was correspondence from Lord Byron, his wife, and their daughter.

The future Lady Lovelace was Byron's only legitimate child. Having been deserted by the wayward and rakish poet within months of their child's birth, Lady Byron perhaps understandably steered her daughter away from the world of poetry, preferring to immerse her in the world of mathematics and science. Lady Byron herself had been a keen mathematician before meeting her husband, who had dubbed her "the Princess of Parallelograms."

Lady Lovelace's propensity for the subject and her family connections in London society led her to meeting and working with some of the most eminent scientists of her day. Among these were Charles Wheatstone, the noted researcher into optics and acoustics, and Babbage. Lady Lovelace had occasionally written to Babbage, but it was Wheatstone who first led her to become closely involved with Babbage's work.

In 1840, Babbage had been invited to a scientific convention in Turin, Italy, which had resulted in an account of his work being written by the Italian military engineer Luigi

Menabrea. A French version of Menabrea's paper appeared in *Bibliotheque Universelle de Geneve* in 1842. Wheatstone (who worked for the British periodical *Taylor's Scientific Memoirs*, which specialized in publishing translations of articles from foreign journals) spotted this article. Wheatstone offered the translation job to Lady Lovelace, and so her association with Babbage began. When Babbage heard that Lady Lovelace was working on the paper, he invited her to add her own notes. She did so to great effect, though the subsequent course of her work with Babbage was never an easy one.

One of the newly found letters is from Lady Byron to her daughter, dated 13 November (no year is quoted), and hints at the Byronic side of Lady Lovelace's character, which could occasionally lead to her scientific thoughts being infused with metaphysical overtones. Lady Byron obviously saw it as her duty to steer her daughter away from her more extreme flights of fancy and to stay on a strictly mathematical course:

> Dearest Ada—It is my part to be a calm observer of your courses, and to employ my sympathies in estimating rather than exalting you. You gave an admirable account of duty when you said that it consisted in your "putting and maintaining yourself in such a state, physical and mental, that God and his agents could use you" & c.
>
> The *Hypothesis* that you [illegible] a "Prophecy" in any sense beyond that which all intellectual beings may claim it, *is to be established* by proofs of your own insight into the natural and spiritual world—These must result from your union with the "All-knowing Integral"—from whom may nothing divert you!

However, as commentators on her work have since pointed out, Lady Lovelace's imagination enabled her to see beyond the technical minutiae of Babbage's calculations and to perceive the greater potential of his machines. Her comparison of the "algebraic patterns" computed by the analytical engine to the fabric patterns woven by a Jacquard loom is a vivid metaphor decades ahead of its time.

Three of Lady Lovelace's letters addressed to Babbage provide some new insight into their often uneasy working relationship. One letter refers to her involvement in the development of what are referred to as "variable cards" for use with Babbage's machines:

> I have just received your letter about the Variable Cards and I hasten to send you these few lines which will be put into the twopenny post for you in Town this morning by means of an accidental opportunity.
>
> I perceive nothing in what I sent yesterday, which is at all inconsistent with the explanations you gave. This is lucky. I hope that you may receive this in time to send me back the sheet I sent you yesterday, from which I wish to make one very trifling alteration in a part of it. And I can only do so myself. If it is gone, I must manage it in the proofs in which I suppose I could insert a sentence.

The letter reflects Lady Lovelace's close level of involvement in Babbage's work and later hints at his abrupt and peremptory attitude in making corrections to her work, as she ends the letter: "I hope you unpasted what you had so cruelly eclipsed."

Indeed, her professional relationship with Babbage seems at times to have been rather fraught, especially when it overlapped with her personal life. On one occasion, Babbage had advised Lady Lovelace's husband (William King, the Earl of Lovelace and also Baron Ockham) against taking a seat on the board of a railway company. Lady Lovelace wrote to Babbage, incandescent with rage:

> My dear Babbage—You cannot *conceive* the mischief you have done me by dissuading Lovelace from taking part in the proposed Central Railway scheme. It is the very thing which my mother & I had looked for him as an occupation calculated to occupy his restless mind which needs work and occupation.
>
> To my surprise and extreme distress, he has just come in saying that seeing *you, you have pointed out to him he is not a man who can need an occupation*!
>
> The least you can do for us after this mischief you have done is to suffer something in the place of what as I tell you had been looked to a Godsend for our family quiet—you can have no conception of what my husband is like when his home alone occupies his irritable energies. So remember this.

On a more somber note, another letter to Babbage conveys a vivid picture of the suffering Lady Lovelace endured due to cancer of the uterus, which was to lead to her death at 37:

> My dear Babbage—It would be a pleasure for me to see you this evening, even if only for half an hour; but as long as you like would be preferable. Could you call in about eight or nine. I have been very ill *really*, & confined to my bed for some time.

Further Reading

B. Wooley, *The Bride of Science: Romance, Reason, and Byron's Daughter*. London: Macmillan, 1999.

D. Stein, *Ada: A Life and a Legacy.* Cambridge, Mass.: MIT Press, 1985.

> It has been impossible for me to leave Ockham now for many weeks; as I have only now come (yesterday) to feel myself a little ... I have been desperately ill. I never had anything of *this*. I have escaped with my life.

Without saying what she wished to see Babbage about, the letter ends.

Lady Lovelace's contribution to the early years of computing science was acknowledged and commemorated in 1980, when the U.S. Department of Defense named the programming language for its military systems after her: Ada.

Christopher Goulding is a PhD research student in the Department of English Literature at the University of Newcastle upon Tyne.

Christopher Goulding
39 Ashleigh Grove, Jesmond
Newcastle upon Tyne NE2 3DJ, England

References

1. M.A. Jackson, *Principles of Program Design*. London: Academic Press, 1975.
2. M. András, *Programtervezés Jackson-módzerrel.* Budapest: Computing Applications and Service Company, 1983.
3. R.S. Burgess, *An Introduction to Program Design Using JSP*. London: Hutchinson, 1984.
4. J.R. Cameron, *JSP & JSD: The Jackson Approach to Software.* Washington, D.C.: IEEE CS Press, 1983.
5. J.R. Cameron, "An Overview of JSD," *IEEE Trans. Software Eng.*, vol. 12, no. 2, pp. 222–240, Feb. 1986.
6. J.R. Cameron, "The Modelling Phase of JSD," *Information and Software Technology*, vol. 30, no. 6, pp. 373–383, July/Aug. 1988.
7. J.W. Hughes, "A Formalisation and Explication of the Michael Jackson Method of Program Design," *Software Practice and Experience*, vol. 9, pp. 191–202, 1979.
8. L. Ingevaldsson, *JSP: A Practical Method of Program Design.* London: Input-Two-Nine,1979.
9. L. Ingevaldsson, *JSD—metoden for systemutveckling.* Lund, Sweden: Studentlitteratur, 1985.
10. M.A. Jackson, "Information Systems: Modelling, Sequencing, and Transformations," *Proc. Third Int'l Conf. Software Eng.*, Washington, D.C., 1978, pp. 72–81.
11. M.A. Jackson, "Jackson Development Methods: JSP and JSD," J.J Marciniak, ed., *Encyclopaedia of Software Engineering*. New York: John Wiley & Sons, 1994, vol. 1, pp. 585–593.
12. H. Jansen, *Jackson struktureel programmeren.* Arnheim: Academic Service, 1983.
13. M.B. Josephs, C.A.R. Hoare, and H. Jifeng, "A Theory of Asynchronous Processes," Oxford Univ. Computing Laboratory Technical Report PRG-TR-6-1989.
14. J. Kato and Y. Morisawa, "Direct Execution of a JSD Specification," *Proc. COMPSAC*, Washington, D.C., 1987.
15. K. Kilberth, *Einfuhrung in die Methode des Jackson Structured Programming.* Braunschweig, Germany: Vieweg & Sohn,1988.
16. D. King, *Creating Effective Software: Computer Program Design Using the Jackson Methodology.* Yourdon Press, 1988.
17. C.D. Poo and P.J. Layzell, "Enhancing the Software Maintenance Factor in JSD Using Rules," *Proc. CompEuro*, 1990, pp. 218–224.
18. C. Potts, A. Bartlett, B. Cherrie, and R. McLean, "Discrete Event Simulation as a Means of Validating JSD Specifications," *Proc. 8th ICSE*, 1985.
19. B. Sanden, *Systems Programming with JSP*. Bromley, England: Chartwell-Bratt, 1985.
20. K.T. Sridhar and C.A.R. Hoare, "JSD Expressed in CSP," Oxford Univ. Computing Laboratory Technical Monograph, PRG-51, 1985.
21. A. Sutcliffe, *Jackson System Development*. London: Prentice-Hall International, 1988.
22. J.B. Thompson, *Structured Programming with COBOL and JSP.* Bromley, England: Chartwell-Bratt, 1989.
23. M.A. Jackson, *System Development*. London: Prentice-Hall International, 1983.
24. J.R. Harris, "The Earliest Solid-State Digital Computers," *Annals of the History of Computing*, vol. 21, no. 4, pp. 49–54.
25. L.C. Brown, "Flyable TRADIC: The First Airborne Transistorized Digital Computer," *Annals of the History of Computing*, vol. 21, no. 4, pp. 55–61.
26. *IRE Trans.*, vol. EC-4, no. 3, p. 101, Sept. 1955.