IEEE Annals of Software Engineering Volume 22 Number 2, pages 61-63, 66, April-June 2000

The Origins of JSP and JSD: a Personal Recollection

by Michael Jackson

My first serious programming work was done 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 realised that program design was hard and the results likely to be erroneous. Into the Honeywell programs, which formed a little system for an extremely complex payroll, I wrote some assertions, with run-time tests that halted program execution during production runs. Time constraints didn't 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, all this could be done directly from the console typewriter. After several weeks without halts, there seemed to be no more errors. Before leaving the organisation, I replaced the run-time halts by 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 consultancy 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 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; but they didn't 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 mutually incompatible, but we weren't sure how to handle this difficulty. We also recognised — it was Barry's idea — that backtracking was an important technique, seemingly ignored by everyone else, practitioner or academic, working on program design methodology.

A vital stimulus to our work on program design was a project — 'the Microsystem' — that 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 on-line transaction-based solution was not economically practical at the time; and 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. Barry 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 256KBytes) were becoming commonplace and data processing program sizes increased dramatically. As programs grew larger, monolithic program texts became less and less practical. 'Modular Programming' became a widely recognised concern, and the first shoots of the 'Structured Revolution' began to appear. Larry Constantine ran a Modular Programming Symposium in 1968, with contributions from Constantine himself and from George Mealy, along with several others. Barry and I found an audience for seminars on our program design ideas, both in the UK and in the USA.

At the end of 1970 I left Hoskyns and started my own 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 US for the ACM, and then a series of one-week courses in the UK, 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 semi-coroutines 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 systematised. And 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 East. In 1974 the UK government adopted JSP (under the name 'SDM') as its standard program design method. Brian 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.

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

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. JSD was based on ideas that had been put forward in 1975, in Principles of Program Design. An information system could be seen as a simulation or model of the 'real world', with added functionality to provide the information outputs. The real world was viewed 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 which 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 coherent 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. The method was described in the book System Development, published in 1983, and in the same year John 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 recognise 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 problem and system. 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. John 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 they were less successful than the COBOL preprocessor and other JSP tools developed by our Swedish licensees and by the Atomic Energy Research Establishment at Harwell.

Over the years from 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, in 1999, JSP and JSD are still in use in Europe and the US, but in a relatively small number of organisations. New tools are still being built.

Acknowledgements

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.

Bibliography

[András 1983] Merély András; Programtervezés Jackson-módzerrel; Computing Applications and Service Company, Budapest; 1983 (in Hungarian).

[Burgess 1984] R S Burgess; An Introduction to Program Design Using JSP; Hutchinson, London; 1984.

[Cameron 1983/89] J R Cameron; JSP & JSD: The Jackson Approach to Software Development (1st edition 1983, 2nd edition 1989); IEEE CS Press, Washington DC; 1983, 1989.

[Cameron 1986] J R Cameron; An Overview of JSD; IEEE Transactions on Software Engineering, Volume 12 Number 2; pp 222-240; February 1986. (Reprinted in Cameron 1983/89, 2nd edition)

[Cameron 1988] J R Cameron; The Modelling Phase of JSD; Information and Software Technology, Volume 30 Number 6; pp 373-383; July/August 1988.

[Hughes 1979] 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.

[Ingevaldsson 1979] Leif Ingevaldsson; JSP: A Practical Method of Program Design; Input-Two-Nine, London; 1979.

[Ingevaldsson 1985] Leif Ingevaldsson; JSD — metoden for systemutveckling; Studentlitteratur, Lund; 1985 (in Swedish).

[Jackson 1975] M A Jackson; Principles of Program Design; Academic Press, London; 1975. (Also translations in Dutch, German, Japanese, Portuguese, and Spanish.)

[Jackson 1976] M A Jackson; Constructive Methods of Program Design; in Proc 1st ECI Conference 1976, pp 236-262; Springer Verlag LNCS 44, Heidelberg 1976. (Reprinted in Cameron 1983/89)

[Jackson 1978] M A Jackson; Information Systems: Modelling, Sequencing, and Transformations; in Proc 3rd International Conference on Software Engineering, pp 72-81; IEEE CS Press, Washington DC; 1978.

[Jackson 1983] M A Jackson; System Development; Prentice-Hall International, London; 1983. (Also translations in Dutch and Japanese.)

[Jackson 1994] Michael Jackson; Jackson Development Methods: JSP and JSD; in Encyclopaedia of Software Engineering, John J Marciniak ed, Vol I pp 585-593; John Wiley & Sons, 1994.

[Jansen 1983] Henk Jansen; Jackson struktureel programmeren; Academic Service, Arnhem; 1983 (in Dutch).

[Josephs 1989] Mark B Josephs, C A R Hoare and He Jifeng; A Theory of Asynchronous Processes; Oxford University Computing Laboratory Technical Report PRG-TR-6-1989.

[Kato and Morisawa 1987] J Kato and Y Morisawa; Direct Execution of a JSD Specification; in Proc COMPSAC 11; IEEE CS Press, Washington DC; 1987.

[Kilberth 1988] Klaus Kilberth; Einf"uhrung in die Methode des Jackson Structured Programming; Vieweg & Sohn, Braunschweig; 1988 (in German).

[King 1988] D King; Creating Effective Software: Computer Program Design Using the Jackson Methodology; Yourdon Press, USA; 1988.

[Poo and Layzell 1990] C-C D Poo and P J Layzell; Enhancing the Software Maintenance Factor in JSD Using Rules; in Proc CompEuro'90, pp 218-224; IEEE CS Press, Washington DC; 1990.

[Potts et al 1985] C Potts, A Bartlett, B Cherrie, and R McLean; Discrete Event Simulation as a Means of Validating JSD Specifications; in Proc 8th ICSE; IEEE CS Press, Washington DC; 1985.

[Sanden 1985] Bo Sanden; Systems Programming with JSP; Chartwell-Bratt, Bromley; 1985.

[Sridhar and Hoare 1985] K T Sridhar and C A R Hoare; JSD Expressed in CSP; Oxford University Computing Laboratory Technical Monograph PRG-51; 1985.

[Sutcliff 1988] A Sutcliffe; Jackson System Development; Prentice-Hall International, London; 1988.

[Thompson 1989] J B Thompson; Structured Programming with COBOL and JSP (2 Vols); Chartwell-Bratt, Bromley; 1989.

27th December 1999