Flow diagrams: Rise and fall of the first software engineering notation

S.J. Morris[1] and O.C.Z. Gotel[2]

[1] Department of Computing, City University, London
sjm@soi.city.ac.uk
[2] Department of Computer Science, Pace University, New York
ogotel@pace.edu

Abstract. Drawings of water are the earliest, least abstract forms of flow diagram. Representations of ideal or generalised sequences for manufacturing or actual paths for materials between machines came next. Subsequently documentation of production and information flow become subjects for graphical representation. A similar level of abstraction was necessary for representations of invisible flows such as electricity. After initial use to define control, flow diagrams became a general purpose tool for planning automated computation at all levels of composition. Proliferation of syntax variants and the need for a common language for documentation were the motivations behind standardisation efforts. Public communication of metalevel systems information superseded private comprehension of detailed algorithmic processes as a primary function. Changes to programming language structures and their associated processes caused the initial demise of flow diagrams in software engineering.

1 Introduction

This paper describes the origins of the flow diagram in engineering practice and charts the development of the underlying ideas as they became integral to early software engineering practice. It is the first in a two-part series examining the impact and legacy of flow conventions in contemporary software engineering diagrams.

In a paper written in 1946 Goldstine and von Neumann provided a succinct statement of a still not fully satisfied need in the field now know as software engineering, "An efficient and transparent logical terminology or symbolism for comprehending and expressing a problem, no matter how involved, in its entirety and in all its parts; and a simple and reliable step-by-step method to translate the problem ... into the code" [1]. A major part of their proposed approach [2] involved the drawing of 'flow diagrams' which came to form the work of all programmers until use of the first generation of higher level programming languages was widespread. The survival of this earliest form of special assistance into the time of manufacturing-like processes makes it one of the most long-lasting craft practices and suggests that its development might indicate typical characteristics for usefulness and longevity in software engineering notations.

Graphical representations of material flows and, at a more abstract level, of process sequences were already commonplace and had fixed conventions by the time that the first electronic computers were being planned. Section 2 provides a selective view of these early manifestations. Section 3 sets out the contribution of von Neumann and Goldstine as described in their publications and uses archival material from the United Kingdom to outline developments in practice under their influence. Section 4 shows how the flow diagram notations were later both formalised in their syntax and altered in their application during the period when their role as a programming aid was diminishing. Section 5 outlines work examining the many progeny for which Goldstine and von Neumann's type of flow diagram is the sole ancestor.

2 Before Goldstine and von Neumann

2.1 Flow in nature

Both as a hydraulic engineer and as a scientist fascinated by the workings of nature Leonardo da Vinci studied the flow of water throughout his life. A folio in the Codex Atlantico [3] shows details of a set of lock gates, part of Leonardo's Milanese work. On the lower part of one sheet there is a view of a weir and a cross-section of the sluice showing lines of motion past it. These drawings and their associated notes are certainly forms of engineering design, whilst others are clearly studies of water itself, always in motion and often impeded in some particular way.

A number of sheets now in the Royal Collection at Windsor Castle [4] all show a clear and rapidly moving body of water passing around an oblong obstacle, shown so that its position in relation to the flow is clear. Views from alternative positions also appear, in one case accompanied by notes comparing the movement to that of hair and describing the flows in terms of "whirling eddies, one part of which is due to the impetus of the principal current and the other to the incidental motion and return flow" [5].

The studies of water flow *per se* made by Leonardo emphasise the lines of movement and the shapes of vortices rather than phenomena such as spray or the play of light. They are illustrative figures which accompany texts and which, without representing exact appearance, give an outline or general scheme of what they represent. As such these drawings, concerned with the material that provides the first paradigm and metaphor for flow, are the earliest and least abstract forms of flow diagram. They are also early evidence of the heavy and continuous dependence of engineering on non-verbal learning and non-verbal understanding [6]. The developments that followed extended flow representations to other materials, man-made forces of movement and industrial processes, higher levels of abstraction and other rhetorical forms.

2.2 Course of manufacture

The primary concern of Leonardo, shared by generations of engineers that followed him, was the harnessing of a natural force, the flow of water induced

by gravitation, either as a source of power or in order to facilitate transport. The industrial revolution which began in the late eighteenth century added a broader interest in the motion of a wide variety of solid and liquid materials, in particular as manufacture transformed raw inputs into finished goods. In this context flow comes to mean, in the words of a textbook on flour manufacture, "the course through which any material travels whilst undergoing manufacture", and this course may be either some ideal and generalised sequence of manufacturing processes or an actual physical path from machine to machine "in their relative positions" or "giving the travel ... in the most direct manner irrespective of vertical or horizontal travel" [7].

Graphical representations of flows in this new manufacturing context normally have the title of 'flow sheet', a term that continued in general use, sometimes hyphenated, until the late 1940s and its supercession by 'flow diagram', by 'flow chart' and finally by 'flowchart'(as both noun and verb). The numbers of processes and machines involved often demanded a more general representation at some higher level of abstraction where the term 'block', or something similar term, came to be used. The same 1912 textbook refers to "care in 'block spacing' the main machines on the space available before filling in the flow lines. The skeleton or block flow-sheets ... emphasize the main masses into which each section of the several processes should be grouped ... the main operations are shown *en bloc*, and will serve as a frame upon which to hang the actual and of course more involved flow-sheet following ...". This notion of an hierarchy of representations, whether implicitly or explicitly defined and whether intended for illustration, or inscription of a design, has carried through to the most recent diagrammatic conventions, including those presented for software engineering.

2.3 Early industrial conventions

Early publications for chemical engineers show how the conventions used for material flows varied widely, particularly in the level of abstraction used in following engineering drawing conventions and in the deployment of arbitrary symbols to denote plant components or processes. Figure 1 shows a typical flow sheet published in 1943 [8] which incorporates standard symbols for a valve (similar to '><'), and for direction of flow (an arrow head), plus simple and complex graphical forms derived from engineering drawings in plan and cross section to denote pipes (single line) and evaporation equipment (labelled '1', '2' and '3').

Drawing derivatives persisted along with purely diagrammatic forms used to show abbreviated and abstracted descriptions. Figure 2 appeared in 1933 [9] to show the main operations within a process, described in detail in the surrounding text, for making pure potassium chlorine. This diagram uses the conventions which remain standard for 'top-down' sequence and layout, and for operations identified by title and isolated within rectangular boxes. These operations also have dual outputs and inputs, e.g. FILTRATION. At this level of abstraction the notion of a fixed sequence of processes becomes as important as any idea of the flow of material.

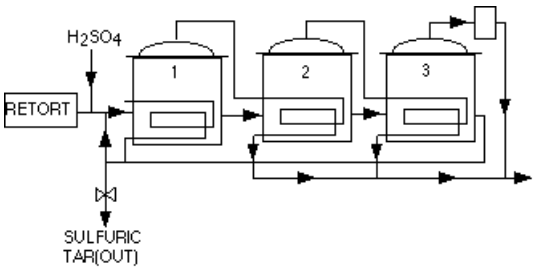

Fig. 1. 'Fig 16 Flow sheet of triple-effect evaporator for removing tars', 1943.

Figure 3, originally also published in 1933 [10], is one of the earliest instances of the use of the term 'flow diagram' applied to a form akin to the current conventions. The text reads "Figure 7 represents preliminary flow sheets for a plant treating 2000 tons of raw polyhalite per day by processes 7 or 7A". In this diagram titles of operation are just underlined and boxes hold details of their outputs. Due to the number of inputs, products, by-products and residues, the top-down convention almost collapses and the direction of each flow arrow appears to be dictated merely by needs of layout. Such diagrams provide a means for presenting in an abbreviated graphical form the many stages in some complex chemical engineering.

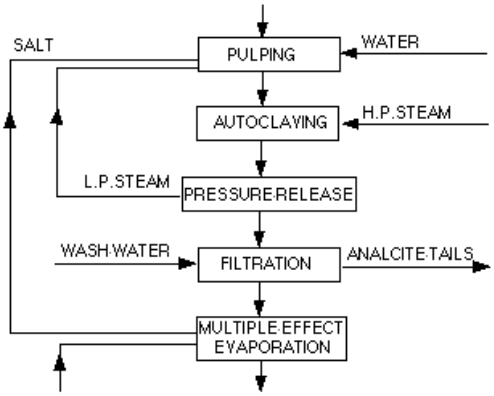

Fig. 2. Section from 'Figure 2. Flow sheet for Production of Potassium Chloride from Wyomingite', 1933.

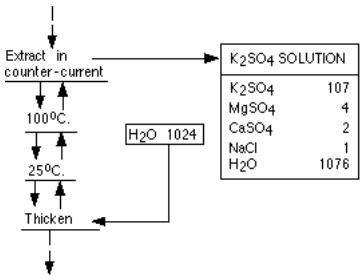

Fig. 3. Section from 'Flow diagram for Processes 7 and 7a', 1933.

2.4 Abstraction of flows

The OED attributes the first use of the term 'flow chart' to a book published in 1920 [11]. Although the figure to which the term is applied [12] is clearly what would now be called a Gantt chart, a diagram type invented at least two years earlier, this publication does illustrate another important source of concepts and forms associated with the idea of flow. This source lies in the field of industrial management where the principal concern is the overall sequence of operations and hence the layout of machines on assembly floors in order to maximise use of machines, in particular by providing a steady supply of materials or partially completed products. Departmental organisation, documentation of production, movement of records, and hence flow of information, become subjects for graphical representation. The book by Knoeppel, as its title suggests, advocates a comprehensive use of graphical representations in production management.

The use of charts in the same book illustrates further abstractions of the concept of flow whereby it may refer additionally not only to the movement of physical documents about materials but also to the information contained in those documents. One figure [13] shows the organisation of a production control department including, with a specific broken line and arrowhead notation, the flows of 'records to accounting dept. for costing purposes', 'various shop controls', 'records', 'statistics' and 'data'. These last two categories represent the highest levels of abstraction represented so far and the only ones not directly denoting some physical entity.

2.5 Invisible flows

A similar level of abstraction had also become necessary for another class of representations where flows are normally invisible. By the mid 1920s the terms 'block schematic', 'schematic diagram' or simply 'schematic' were in common use for any meta-level representation showing the arrangement of electrical systems comprising multiple components but without any use of detailed circuit notation. Figure 4 shows a typical example from 1935 [14]. The simple syntax of rectangular boxes with text labels, lines and arrowheads is the same as that for

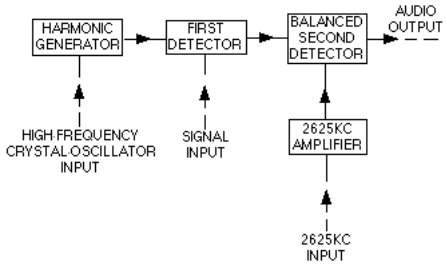

Fig. 4. 'Fig. 3 - Schematic of transmitting monitoring device', 1935.

flow sheets and the arrangement of elements follows no two dimensional rules except the standard western preference for some form of left-to-right sequence. Such general representations of electrical systems would have been familiar to the engineers building and operating the first electronic computers and their use continued as a means for illustrating functional components and communication of early machines. Representations of hardware became an issue in themselves and are not considered here.

During the period prior to the appearance of flow diagrams in the field of computation, the notions of flow and the manner of their denotation became steadily more complex both in terms of the materials and processes involved and the manner of their representation. Regarding what was actually flowing, multiple, visible, synthetic materials, most clearly seen in the chemical industry, came to join simpler, single, natural or manmade materials as the subjects for examination and illustration. Enclosed manufacturing plant normally rendered flows as invisible as electric current. Multiple stage processes generating, manipulating or terminating flows, whether seen or not, steadily increased the level of complexity.

2.6 Symbolic representation and sequence

Although the schematised representations by Leonardo of manipulated water flows show a degree of abstraction from physical resemblance, they are as far from current notions of a flow diagram as are his drawings of machines from standard engineering drawings. In an intervening stage of the development of representations, conventional drawings such as pipework cross-sections derived from the scaled representations intended for construction, could be related to physical resemblance. In simplified forms these symbols acted as icons, in the Piercian sense derived directly from the appearance of the objects to which they referred. Such static representations required the addition of a classic Piercian index, the arrow or arrowhead, to indicate direction of flow and denote dynamic change.

Introduction of arbitrary symbolic elements in the early representations, most importantly the rectangular box, became necessary as the processes of flow be-

came further subdivided and required some higher level description to record all significant components within one overall form. It is clear from the earliest examples that such boxes serve a graphical as much as a denotational purpose, circumscribing within an overall layout each of a number of individual textual labels which themselves do the referring to process definitions. However arrangement of the boxes themselves does contribute an essential element of meaning in so far as it follows some convention for sequence, most importantly top to bottom or left to right.

The notion of sequence is implicit in all the representations of flow considered so far. The later variants incorporate an essential new element of separation between two levels of an hierarchy. Individual elements in one diagram denote abbreviated or abstracted versions of more complex wholes at the level below in a number of other diagrams or forms of representation. In some variants there is also a means of showing multiple inputs and outputs to any component and thus the bifurcation and recycling of material. The notion of a flow diagram now incorporates a process broken down into individual sub-processes in a set sequence, each component denoted by an abbreviated title presented within a box and each stage in the sequence by at least one arrow or arrowed line between boxes representing succeeding and preceding component. This is essentially a meta-level form predicated on some lower and more detailed level of meaning into which the represented process will later be transformed by some means.

3 The GvN type and after

3.1 Flow diagram of machine control

The new flow diagrams of Goldstine and von Neumann (here abbreviated to 'GvN type') altered the basic definition in two fundamental ways. Their domain of operation is now abstract and mathematical rather than material, even at the sub-atom level, and their use is embedded within a staged process which makes use of an hierarchy of representations and is related to specified operational semantics of a computational machine.

Goldstine describes how "an exceedingly crude sort of geometrical drawing" developed from being a means to show the iterative nature of 'an induction' to being an essential means for expressing a mathematical problem and hence for programming [15]. Development of the notation took place incrementally. The 'assertion box' [16] was the last of a series of innovations which altered radically both the syntax and semantics of the flow diagram and defined its paradigmatic form.

The GvN type retained the single essential character from the syntax of its predecessors, the combination of at least two rectangular boxes representing operations of some kind and an arrowed line connecting them to define their sequence. Figure 5 shows the initial published example in which operations boxes are left unlabelled and have single connectors at opposite sides. Preceding types of flow diagram had incorporated operations with more than one input or output but without altering the generic operational meaning.

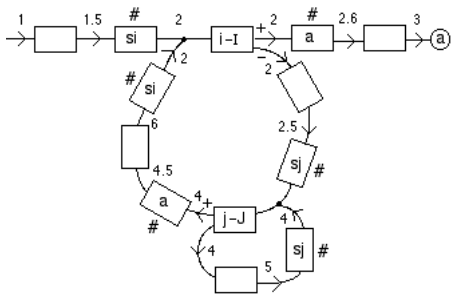

Fig. 5. Initial 'flow diagram' example given by Goldstine and von Neumann, 1947.

The most long-lasting innovation in the GvN type was the introduction an 'alternative box' with 1 input arrow and 2 output arrows. In the Figure 5 diagram the alternative boxes contain the 'induction variables', i and j, and the numbers at which induction ends, I and J. The outputs have plus and minus signs as labels indicating alternative courses if the test indicated within the box produces a positive or negative result. In combination with a simple junction notation to show where connectors converge, the operation and alternative boxes enable representation of single or multiple 'induction loops' and the corresponding iterative process.

3.2 Definition of new semantics

It is the logic of such iterative mathematical processes rather than any flow of data that is the motivation for the GvN type. The diagram is "a schematic of the course of C(ontrol) through the sequence" of code that will be consequently defined. The accompanying text [17] gives precise semantics to the alternative outputs from alternative boxes by mapping them to specific conditional transfer orders (xCc and xCc' defined elsewhere [18]). Clarification of this course required the introduction of two additional types of boxes for 'substitution' and 'assertion', both labelled outside with what is now called a 'hash' (or 'pound') sign. In the simplest case a 'substitution' box provides the means to show that at least one variable, the induction variable, must change before control completes any loop, but specifically not to affect that change. In Figure 6 the loops conclude with boxes labelled 's i' and 's j'; boxes labelled 'a' appear at the exit from each loop after each 'alternative', making clear the 'assertion', for example 'i = I', which is a *de facto* restriction at this point.

The GvN type also incorporates the first specific semantics for the syncategorematic signs which form the connections between box symbols. These symbols dissect the flow diagram into 'constancy intervals', numbered with Arabic numerals, along which all changing variables and storage locations have constant values. A separate box connected by a broken line to the connector provided an optional space (not used in Figure 5 example) within which to record values. All

successive contents of storage positions must also be shown in a separate 'storage table'. This notion of constancy within the interval between defined process elements is present in earlier and later types of flow diagram although neither made explicit nor discussed. Construction of a (usually partial) storage table comes in the second stage of the coding process of which the GvN type forms an integral part.

The drawing of flow diagrams came after mathematical preparations, the choice of suitable equations and conditions, clarification of the explicit procedures involved and estimation of precision. The authors emphasised that this first of four main stages had nothing to do with machines or mechanisation. To use contemporary expressions, the standard mathematical metalanguage already provided a suitable modelling framework.

In the second stage the drawing of flow diagrams provided the means to plan the course of the mechanical computation. This was a 'macroscopic' or 'dynamic' stage of coding [19], accompanied by the drawing of a (normally partial) storage table as the diagram evolved. The 'static' or 'microscopic' stage of coding [20] completing the contents of every operation and alternative box followed. Assignment of all storage positions and final order numbers comprised the fourth and final stage. Omitting the storage activities, these macro and micro stages correspond to levels now often called 'modelling' and 'implementation' in program development. The flow diagram proved to be an extremely successful syntactic form for computational 'models' related to the mathematical problems which were most of interest to many potential users. This remained the case until the market for commercial computers began to burgeon a decade after definition of the GvN type.

3.3 Standard tool for programming

Program documentation that survives from the immediately post-war period in the UK shows that the basic process defined by von Neumann and Goldstine provided the framework for programming the earliest versions of the machine designed initially by Turing at the National Physical Laboratory at Teddington (NPL). An example developed for the ACE Version 8A (the design immediately preceding the Pilot Ace that first ran in May 1950) is a program written by Mike Woodger for the solution of the differential equations for the motion of a dampened string [21]. The first three pages give a general description of the problem, its solution in principle and an elaboration of the method chosen. The fourth page defines 'Subsidiarys' (sub-routines for output and particular storage operations) and 'Storage Arrangements' for the fourteen delay line storage components. The next page consists of a flow diagram drawn on lined and columned paper using three columns on the left for boxes (reproduced in part as Figure 6). There follows a page of detailed code for the 'OUTLET' sub-routines and three pages of detailed code with the heading 'MAIN TABLE IN DETAIL'. The final two sheets show storage allocations in detail.

The intensive use of sub-routines began at an early stage in England [22]. This development did not effect the general function of the flow diagram; it

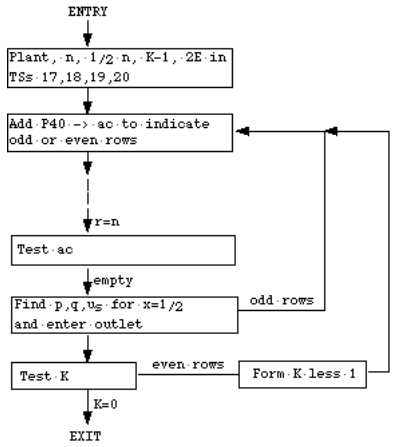

Fig. 6. Flow diagram for ACE 8A programme, 1947/48.

merely made explicit the possibility of an hierarchy of generality in its use. In 1951 Turing used the much older term 'block schematic diagram' for an essential aid in defining subroutines. In Section 15 of his programmers' handbook for the Manchester machine [23], Turing defines 'Programming principles' which presuppose a two level combination of main program and sub-routines.

The principle steps that he recommends comprise 'plan making', 'breaking the problem down' and 'programming subroutines'. The plan should include the apportionment of storage to various duties as as well as decisions about mathematical approach and formulae to be used. The 'block schematic diagram' provides a convenient aid to planning at the subsidiary level of the subroutine. It consists of "a number of operations described in English (or any private notation that the programmer prefers) and joined by arrows. Two arrows may leave a point where a test occurs, or more if a variable transfer number is used. Notes may also be made showing what is tested, or how many times a loop is to be traversed." [24]. After preparing the diagram "the operations appearing as blocks [within it] may be replaced by actual instructions." Finally there is the programming of the "main routine" about which there is simply the advice that it "follows principles similar" to those for the subroutines. The only flow diagram of any kind in this document (other than interesting contemporary additions inserted by Woodger on punched cards) is simple and shows a high level of generality (reproduced as Figure 7).

In all these diagrams the arrow symbol has lost the explicit and more restricted semantics assigned to it in the GvN type and the box symbols with enclosed labels now denote only operations or tests. The notion of flow has reverted to that of the controlled sequence. Flow diagrams have become a general purpose tool for planning an automated computation at all levels of composition or decomposition. The notion of flow has become synonymous with sequence as

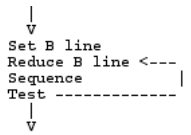

Fig. 7. Untitled diagram from Manchester MkII Handbook, 1951.

issues of computation and storage have been separated and although operations are dealt with one-by-one every machine has unique sets of commands and operations requiring its own sequence of operations to solve particular numerical problems.

Although at a much later date a dedicated advocate of flow charts could write that "neither programmers nor analysts think in terms of flowcharts" and would justify their use by their "documentation value" [25], when the first attempts were being made to program complex mathematical procedures such as the Runge-Kutta method for solving numerically ordinary differential equations, "understanding the process demands the use of flow diagrams" [26]. This comment appears as the first point in a comparison made during 1959 between programs for the integration of differential equations written in the language used on DEUCE (the English Electric Company's commercial version of ACE) and in the newly defined ALGOL language. The complexity of the algorithms being constructed is illustrated by part of Woodger's own flow diagram reproduced in facsimile as Figure 8 [27].

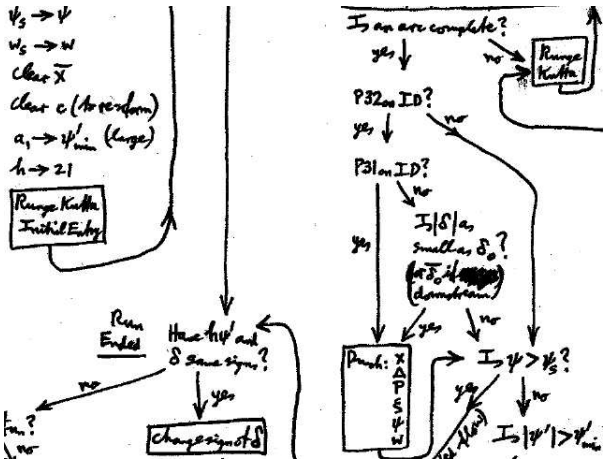

Fig. 8. Detail from flow diagram by M. Woodger for a differential equation algorithm, 1959. NPL Archive. Reproduced by courtesy of the National Museum of Science and Industry, London.

4 Formalisation, diversification and demise

4.1 Expanded use by manufacturers and programmers

During the second half of the 1950s as computers became widely available for commercial use, their manufacturers prepared manuals clearly intended for operators expected to have little familiarity with any form of programming. In the UK Ferranti Limited produced commercial machines in collaboration with Manchester University and beginning with its second machine, the Pegasus first delivered in March 1956, prepared manuals for public use. The Pegasus manual presents an explanation of machine structure and an introduction to all aspects of its programming on the basis of principles easily recognised from earlier practice. It recommends the use of a block diagram as, "a less detailed flow-diagram ... in which whole groups of instructions are briefly described and not written down explicitly" when dealing with "a complicated problem" [28].

The flow-diagram itself "expresses the 'algebra' of the process". It is used as the means of expressing the results from an initial stage concerned with "converting the numerical method into a series of steps ... a small group of mathematical operations". Writing down the actual orders that the machine will have to obey is then "relatively simple once the flow-diagram has been obtained" [29]. Over the next two years the Ferranti literature incorporates a full range of uses for flow diagrams including a representation of wage processing at a high level of generality [30] and detailed examples of complete programs including full code [31].

A widening range of applications for flow diagrams and a rapid proliferation of symbols accompanied the increased commercial use of computers. The first ACM glossary of programming terminology published in 1954 defined 'flow chart' as "graphical representation of a sequence of operations using symbols to represent the operations such as compute, substitute, compare, jump, copy, read, write, etc." [32].

By the time of the most recent ISO document [33], now twenty years old, the number of symbols had increased from the basic post-GvN five (operation, test, flow arrow, stop, start) to forty including allowable variants. These standard symbols are to be combined to build up flowcharts at three levels of generality. Data flowcharts and program flowcharts show data flow and control flow at a basic level. System flowcharts show data flow at a higher level at which operations and their control are incorporated. At the same higher level the program network chart shows the path of program activations and interactions with related data. At the highest level the systems resource chart represents the configuration of 'data units' and 'process units'.

Proliferation of syntax variants, both by equipment makers and by major customers doing their own programming, and the need for a generally agreed language for documentation, were the motivations behind the standardisation efforts. Public communication of metalevel information about systems had come to supersede private comprehension of detailed algorithmic processes as the primary function of the flow diagram.

4.2 An alternative semantics

Provision of a means for public communication of numerical methods and procedures and for realizing a stated process on a variety of machines were the first two objectives for the definition of ALGOL [34]. This development prompted the only semantic definition of a flow diagram wholly separate from algorithm or system analysis. The syntactic flow chart for ALGOL [35] contains 328 symbols in all, labelled circles, ellipses and rectangles located within a space with positions defined vertically and horizontally. The sequence function represented by any arrowed line in the standard post-GvN form is now denoted only by the horizontal; the vertical denotes a definition of possible alternatives. This form of chart also incorporates recursion; a rectangular box indicates that the enclosed label refers to an element defined elsewhere at the given co-ordinates of the chart.

The definition of Algol came at the same time as a recognition that the problems then being addressed were those of "dealing not with computers but with general information transformers or symbol manipulators" [36]. Alternative data structures, firstly the 'linked lists' used in LISP and later the 'classes' of SIMULA67, were under development as the spaghetti-like characteristics of sequential programmes incorporating 'go to' statements were becoming a subject of controversy [37].

4.3 Conclusion

In the early days of programming with low-level machine and assembly languages, flow diagrams were obviously at a higher level of abstraction than program code and were routinely used to define the logic of algorithmic processes. The syntax and semantics of the notation had to accommodate the fact that data was stored and manipulated physically and processing was sequential. Flow diagrams emphasised flow of control, the fundamental requirement, and provided a comprehensive representation of a program. With changes in hardware, the development of modern operating systems and the introduction of higher-level languages, all converging during the 1960s, the complexity of the problems tackled could grow. Focus transferred away from the logical details of algorithmic processes and towards the broader concerns of software engineering, which emerged as a separate discipline following discussions at the NATO conference of 1968 [38].

As structured development gained favour in the 1970s and 1980s, flow diagrams came to be used to model the logic of workflow and structured flowcharts [39] were introduced to add discipline to the design of lower-level code. With the move to object-oriented analysis and design in the 1980s and into the 1990s there was even less need to express detailed algorithmic processing as part of routine practice. As software engineering tools flow diagrams of all varieties fell even further out of favour until the advent of the UML in the 1990s and the first introduction of its activity diagrams [40].

The GvN type flow diagram is the sole ancestor of an extended family of software engineering notations and diagrams. This continuing development of

the concept of flow in software engineering and the many resulting notations and diagrams are the subject of further work.

References

1. Goldstine, H.H. and von Neumann, J. *On the principles of large scale computing machines.* In A.H.Traub (Ed.) John von Neumann, Collected Works Volume V, Design of computers, theory of automata and numerical analysis. Pergamon Press, Oxford, 1963, p30.
2. Goldstine, H.H. and von Neumann, J. *Planning and coding of problems for an electronic computing instrument*, Part II, Volume 1. Report prepared for US Army Ordnance Department, April 1947. In A.H. Traub (Ed.) 1963, pp 80-151.
3. *Il Codice Atlantico di Leonardo da Vinci nella Biblioteca Ambrosiana di Milano*, Editore Milano Hoepli, 1894-1904 (656a). Also `http://www.museoscienza.org/english/leonardo/portellochiusa_d.jpg`, January 2006.
4. Clark, K. and Pedretti, C. *The Drawings of Leonardo da Vinci in the Collection of H.M. The Queen at Windsor Castle* (2nd edition), London, Phaidon, 1968 (12579, 12659-12663) Also `http://www.geocities.com/davincigateway/oldmanwater.14.jpg`, January 2006.
5. Richter, J.P. *The Literary Works of Leonardo da Vinci* (2nd Edition), Oxford, 1939.
6. Ferguson, E.S. *Engineering and the Mind's Eye.* Cambridge, MIT Press, 1992, p*xi.*
7. Amos, P.A. *Processes of flour manufacture.* London, Longmans Green, 1912, pp171-178.
8. *Industrial and Chemical Engineering*, American Chemical Society, Volume 35, 1943, p295.
9. *Industrial and Chemical Engineering*, American Chemical Society, Volume 25, 1933, p375.
10. *ibid* p1159.
11. *Oxford English Dictionary.* Online edition at `http://www.oed.com`, August 2004.
12. Knoeppel, C.E. *Graphic Production Control*, The Engineering Magazine Company, New York, 1920, p135.
13. *ibid* p126.
14. *Proceedings of the Institute of Radio Engineers*, Volume 23, Number 7, p706.
15. Goldstine, H.H. *The Computer from Pascal to von Neumann.* Princeton, NJ, 1972, p266.
16. *ibid* p267.
17. *Planning and coding problems*, p87.
18. Burks, A.W., Goldstine, H.H. and von Neumann, J. *Preliminary discussion of the logical design of an electronic computing instrument*, Part I, Volume 1. Report prepared for US Army Ordnance Department, Second Edition, September 1947, p75. In A.H. Traub (Ed.)1963, pp34-79.
19. *Planning and coding problems*, p100.
20. *Planning and coding problems*, p101.
21. NPL Archive, National Museum of Science and Industry, London, N30/30 *ACE Version 8A Description, index, form of instruction word. Seven programs: 2 problems of Wilkinson, INORD, Laplace's equation of relaxation, forward integration, method of characteristics for vibrating string, interpolation 1947/48.*

22. Wilkinson, J.H. Turing's work at the NPL and the construction of Pilot Ace, DEUCE and ACE. In Metopolis et al., *A History of Computing in the Twentieth Century.* New York, Academic Press, 1980, p105.
23. NPL Archive N26/38 *Programmers' Handbook for Manchester Electronic Computer Mark II*, March 1951. Also in facsimile at `http://www.turingarchive.org/browse.php/B/32`, January 2006.
24. *ibid* p64.
25. Chapin, N. Flowcharting with the ANSI standard. A tutorial. *Computing Surveys*, Volume 2, Number 2, pp119-146, June 1970, p143.
26. NPL Archive N9 *December 1959 Points arising from the example of program 4532(6) in Algol.*
27. NPL Archive N9 *File: Simplified programming systems for ACE. M. Woodger work 1959. Incl. complete DEUCE prog. for shockwave boundary layer interaction, Ma4532, 1957-59.*
28. Ferranti Limited. *Ferranti Pegasus Computer Programming Manual Issue 1* List CS 50 September 1955, p1.7.
29. *ibid* p1.6.
30. Ferranti Limited. *Ferranti Pegasus Computer A description of a wage programme for a large dispersed organisation* List CS 169 September 1957.
31. For example Ferranti Limited. *Ferranti Pegasus Computer A simple programme 'Special Factorize'* List CS 153 June 1957.
32. Bright, H.S. Proposed standard flow chart symbols. *Communications of the ACM*, Volume 2, Number 10, October 1959, p17.
33. International Standards Organisation. *ISO 5807-1985 Information processing - Documentation symbols for data, program and system flowcharts, program network network charts and system resource charts.*
34. Backus, J. *The syntax and semantics of the proposed international algorithmic language of the Zurich ACM-GAMM conference*, p129. In Proceedings of the International Conference on Information Processing, UNESCO, Paris, June 1959, UNESCO, Paris, 1960.
35. Taylor, W., Turner, L. and Waychoff, R. A syntactical chart of Algol 60. *Communications of the ACM*, Volume 4, Number 9, p393, September 1961.
36. Gorn, S. *Common symbolic language for computers. Introductory speech.* In Proceedings of the International Conference on Information Processing, UNESCO, Paris, June 1959, UNESCO, Paris, 1960, p117.
37. Dykstra, E.W. Letters to the Editor: go to statement considered harmful. *Communications of the ACM*, Volume 11, Number 3, March 1968.
38. Naur, P. and Randell, B. (Eds) Software Engineering, Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968, NATO, January 1969.
39. Nassi, I. and Shneiderman, B., Flowchart Techniques for Structured Programming, *SIGPLAN Notices*, Volume 8, Number 8, August 1973.
40. Booch, G., Jacobson, I., Rumbaugh, J. The Unified Modeling Language for Object-Oriented Development, Documentation Set Version 0.91 Addendum, Rational Software Corporation, Santa Clara CA, September 1996 pp27-30.