# HUMAN FACTORS IN VIRTUAL MACHINE HIERARCHIES

B.R.Gaines

These notes are a distillation of some basic considerations from diverse experiences, reported in detail elsewhere, in: minicomputer design, both for compact program encoding[1] and for high-level language support[2,3,4]; language design for interactive systems[5,6,7]; programming interactive dialogue[5,8]; and analysing the foundations of structured programming[9], protection structures[10], and future trends in computers[11].

At any one time hardware, software, or applications aspects of computers may seem to predominate, either as bottlenecks to progress, or as offering exciting new possibilities - one may feel that architecture, languages or systems require particular effort, or offer the possibility of a 'breakthrough'. What has become increasingly apparent, however, is the similarity of problems and techniques in these seemingly diverse areas, and the high degree of structural similarity at different levels of computer systems design - making the old hardware/software distinction misleading. The spread of dynamically microprogrammed architectures has given a software flavour to what was previously hardware design. Conversely, the spread of structured programming concepts has transferred hardware techniques of isolation, quality control, modularity, etc., to software engineering.

There is a paradox here in that, whilst hardware and software have become increasingly related at a conceptual level, the gulf between actual gate-level hardware and high-level languages, let alone applications programs, has widened in recent years. Whilst one initial trend in using low-cost l.s.i. was to more closely support the run-time systems of major languages, the overall trend in minicomputers is now towards multilevel microprogrammed structures that are not orientated to a specific language but offer immense flexibility at what used to be the hardware design level, i.e. the deferred-design inherent in computers is being taken even further. This is partly because no current languages are universally adequate and partly because of the need to provide effective emulation of a variety of machines. Thus the use of the term 'system' rather than 'hardware' in the title of this colloquium represents a definite and necessary trend.

The irrelevance of hardware progress in its own right has been exacerbated by the halving of hardware costs over the past three years whilst software, and overall system, costs have doubled. Configurations may be purchased for under £10K that have an immense potential in commercial applications, yet many more tens of thousands must be spent to realize that potential in any particular application. Efforts to relate hardware facilities to standard languages have little effect here because most of the cost is in system analysis and ongoing system development. Thus, in considering 'high-level' languages, we must broaden the scope beyond the conventional ALGOL/CORAL level towards 'library rout-

B.R.Gaines is with the Department of Electrical Engineering Science, University of Essex, Colchester, Essex, UK.

ines', 'packages', 'programmed interaction', and so on. Indeed, it is worth noting that formal languages are not unique to computers but have always played important roles in system design and operation. The expression of circuit designs as drawing office blueprints, the communication of project schedules as bar-charts, etc., all use 'formal languages' that can be manipulated algorithmically.

In considering the diverse levels of language from the microprogram to applications dialogue, the concept of a virtual machine hierarchy is extremely useful. At each level in the hierarchy there is a programming problem in one direction and the provision of programmable resources in the other. One cannot make completely inflexible even the package supplied to the end user - he needs capabilities to add new record structures, printouts, etc. On the other hand one solves no problems by giving total flexibility at any level in the hierarchy. The problem of comprehension and the decision-making and responsibility of the use of excessive flexibility are themselves sources of difficulty. When several users or a project team are involved then excessive flexibility allows for unnecessary diversity of approaches so that modules and documentation are duplicated and interfacing modules becomes difficult.

Thus one should envisage far more levels and structure in the virtual machine hierarchy than are usually considered. Also one must take into account the human factors at each level of the hierarchy. There is a technical relationship between problem and (virtual) machine, e.g. if character-string analysis is required SNOBOL may provide a suitable 'machine'. However, the problem has to be expressed for the machine by a programmer, and that person also has a relationship to both machine and problem, e.g. does he understand SNOBOL, does he think of the problem as one of character strings or one of lists. The presentation shows how the virtual machine hierarchy appears when this 3-part relationship between programmer, problem and machine is incorporated. It indicates: the richness and diversity of the various levels of computer-based system design; the role of human factors in system considerations; the role of language at each level; the decoupling between the levels in a clean virtual machine implemenation; and the iteration of the same basic structure at different levels, giving some hope for overall technology transfer.

References


1. MINIC I Programmers Manual, Micro-Computer Systems, Woking.
2. GAINES, B.R., FACEY, P.V., WILLIAMSON, F.K., and MAINE, J.A., Design Objectives for a Descriptor-Organised Minicomputer, Proc.EUROCOMP 74, London, May 1974, pp.155-169.
3. WILLIAMSON, F K., GAINES, B.R., MAINE, J.A., and FACEY, P.V., A High-Level Minicomputer, Proc.IFIP Congress, Stockholm, August 1974, pp.44-48.
4. GAINES, B.R., HAYNES, M., and HILL, D., Integration of Protection and Procedures in a High-Level Minicomputer, Proc. 1974 Computer Systems and Technology Conference, IEE, London, October 1974.
5. GAINES, B.R., and FACEY, P.V., Some Experience in Interactive System Development and Application, Proc.IEEE, 63(6) June 1975, pp.894-911.
6. FACEY, P.V., and GAINES, B.R., Real-time System Design under an Emulator Embedded in a High-Level Language, Proc.BCS DATAFAIR 73, Nottingham, April 1973, pp.285-291.
7. GAINES, B.R., GEDYE, J.L., and FACEY, P.V., A Versatile Multi-User Interactive Language System for a Minicomputer, Proc.BCS DATAFAIR 71, Nottingham, March 1971.
8. GAINES, B.R., FACEY, P.V., and SAMS, J., An Interactive Display-Based System for Gilt-Edged Security Broking, Proc.EUROCOMP 74, London, May 1974, pp.155-169.
9. GAINES, B.R., Analogy Categories, Virtual Machines and Structured Programming, Proc. 5th Annual Congress of th Gesellschaft fur Informatik, Dortmund, October 1975, in Goos, G., and Hartmanis, J., GI - 5.Jahrestagung, Lecture Notes in Computer Science, No.34, Berlin: Springer-Verlag, pp.691-699.
10. KOHOUT, L., and GAINES, B.R., The Logic of Protection, ibid. pp.736-751.
11. GAINES, B.R., Computer Technology and its Utilization: Today and Tomorrow, Proc. National Engineering Laboratory Conference, 'Small Computer Applications in Industry', East Kilbride, March 1973.