A Philosophy for the Teaching of Computer Science and Information Technology

L. CAPPER
School of Information Sciences, the Hatfield Polytechnic, Hatfield, Herts

Much adverse criticism, most of it informal, has been directed at programmes of study in Computer Science. Part of that criticism has been concerned with the apparent lack of theory and a unifying philosophy within the subject. This paper, presenting the views of the author, proposes that the discipline, whatever its title, can be described as having two central unifying themes. One is based upon a study of data, information and knowledge at different levels of complexity or abstraction, and from which the other areas of the discipline can be developed. The second theme is developed from the premise that Computer Science is essentially a technology concerned with the development and use of particular types of human-created artefacts.

Received February 1984

1. INTRODUCTION

Because of the changing technology, hardware and software, and the growing experience in their use and teaching, it has become necessary to develop and update our curricula. But before proceeding it is essential to decide what academic strategy or strategies to employ in this process. Is it enough just to tinker with the earlier proposals or is a fundamental rethink necessary? Jones & Loomes[1] suggest that four possible strategies, or a mixture of more than one, can be used, as follows:

- input/output education process; decide on what the product should be and, from that specification, design the education process given an expected intake.
- existing scheme modification; criticise the existing programme and attempt to modify the weaknesses.
- what is Computer Science? Specify the essential characteristics of the discipline, and from that decide what are the concepts, domain of knowledge, boundaries and interfaces with other disciplines and the essential skills. From this detailed specification a total programme can be developed.
- central core; identify a central core which is fundamental to the discipline and which can be expanded, while still emphasising the central theme, in the development of a programme.

These strategies are not mutually exclusive and are designed to give direction to any planning committee and/or individuals.

The first directive is in the form of a question, namely, is it necessary to put and answer certain fundamental and searching questions about the nature of the discipline, what it is and, because of the major developments that may have occurred, if a major rethink is necessary. If that is the situation, then tinkering with existing schemes or using existing ideas on the input/output strategy will not produce a viable proposal.

Undoubtedly major changes have occurred in the last decade, in the technology, in our understanding of fundamentals and in their application. One has only to look at the developments in the use of microcomputers and text processing, for example, to realise this and therefore to decide that a major rethink is necessary. The author, because of earlier work,[9] selected the fourth strategy. But to establish a central core the question 'What is Computer Science?' had to be explored, if not initially, at least in parallel and iteratively.

2. THE BACKGROUND CRITICISM

In a half-remembered broadcast of some few years ago, of the type used to fill the interval between the two halves of a concert, an equally forgotten speaker discussed different approaches taken over the years in courses of higher education, describing in the process some of the differences between a good and a great course. Of the relevant factors, the need for a central unifying philosophy was emphasised and illustrated by a number of examples, one of which was an undergraduate programme read by the writer himself. A highly respected and successful course in Zoology run under the leadership of the then head of department, Professor Sir James Gray, used Evolution to provide the basic, central and unifying theme and bring cohesion to this traditional, yet diverse and expanding discipline.

A further example referred to what is possibly the most diverse of sciences, encompassing the greatest volume of knowledge in one discipline, namely Chemistry, and how an approach based upon reaction mechanisms was successful.

This need for a central unifying theme and philosophy has been stressed by many people and organisations. In Britain the degree-validating body for Polytechnics and Colleges, the CNAA (Council for National Academic Awards), has stated this formally in publications,[2] private reports to individual institutes and discussions.

A lack of any coherent theory and philosophy has been used to argue that Computer Science is not a single discipline worthy of undergraduate study and it has even been suggested that Computing is little more than programming.[3] Following upon the largely informal adverse comments about Computer Science, it being a hotch-potch of techniques and experiences with few theories and no philosophy and suitable only for training courses, the writer asked, informally, a number of Computer Science lecturers at a 1981 Cambridge Conference (see Section 3 below) if they would advise their own teenage children to study the subject for their first degree. None said they would recommend it, as such programmes have limited educational value. They preferred Maths, Economics and Physics, for example, followed possibly by a suitable MSc in Computer Science or, probably better, good industrial training. As a career, it was approved. On being asked to expand, some mentioned the strong tendency to lecture at the level of the state of the art, to follow trends, and to ignore even the few underlying theories and principles that do exist. Thus with a rapidly developing technology that has to be applied, the graduates soon find their knowledge obsolete and, because of their poor education, are unable to update themselves and apply new knowledge as easily as they should. The oft-quoted views of employers that graduates in Computer Science do no better in their chosen field after the first few years than graduates in other disciplines have been used to support these theories. Do Computer Science programmes train for the short term rather than educate for the long term? The ACM,[4,5,6] IFIP[7,8] and the BCS[9] curriculum working parties have presented proposals that contain more than just a selection of useful and relevant courses, but a coherent whole. Unfortunately these recommendations go no further in trying to produce a philosophy upon which this rapidly changing and diverse subject of computing, which contains elements of engineering, social and financial studies, science and humanities, can be based. This paper suggests that such a philosophy can be developed based upon a central theme of 'Knowledge, Information and Data'.

3. DEVELOPMENT OF THE PHILOSOPHY

Following on from the annual working conference of the Information Systems Teachers of the UK in July 1981 at Cambridge University, and as described earlier by the author,[10,11] a working group made up of two subgroups was established to investigate a number of questions. One subgroup examined the material to be taught; the second, under the chairmanship of the writer, investigated the possible impact of the changing technology and environment upon what is taught. The work and reports[12–14] of this second group provided one basis for the development of this paper.

Against a background of criticism, increasing change and economic restraint, the first discussions established a need for a coherent programme with a unifying theme, and a platform from which graduates can update themselves as easily as possible. This argument led to the specification of a central core of material, the statement of which posed the question 'Why has this material been selected?'. In answering that question it was possible to identify the underlying philosophy that had resulted in this selection; cause and effect. With iteration the theme was developed.

Reduced to its simplest level, computers and all other electronic devices are essentially machines that when programmed accept, move, process and store data, and output information using some form of symbols, signs and languages according to a particular set of paradigms and algorithms. But what are data? What is information? Apart from saying that they are a representation or map of reality, the answers given to these questions will depend upon the view of the individual and the level of abstraction at which he or she is working.

Perhaps a good way to approach these questions, and show the central part that data/information play in Computer Science, is by using the hierarchical principle – or philosophy – of levels for the real-world complexity of knowledge. This concept describes the knowledge we have in terms of 'levels of complexity', or levels of abstraction, with particular laws, properties, characteristics, activities, etc., appearing or 'Emerging' at only one level. For example, the peculiar properties of biological systems do not appear at the the more fundamental level of Chemistry, although they may be explained in terms of chemical and physical phenomena. Such a hierarchy is, for example,[15] Sociology, Psychology, Biology, Biochemistry, Chemistry, Physics. Not everybody may agree with this list, but that is not important to us. It is suggested that a relevant hierarchy can be developed from one generally accepted approach to the design of Information Systems, namely that of levels of design. Two examples are:

Information Level
Conceptual Level
Physical Level

and

Total Systems Level
Information Level
Data Level
Physical Representation Level.

Similarly, with the development of levels of protocols for data transmission such a hierarchy, a finer hierarchy, can be seen. In an analysis of what is meant by Data and Information we have, at the most fundamental level, the study of electronics (and fibre optics). This study can be used to explain at a higher level of complexity how a digital computer can create and handle bits. This in turn can be used to explain how bits can be employed to encode characters and groups of characters in the form of bytes and words, groups of characters enabling signs to be represented. A study of signs[16] is important if an understanding of data, and hence of information and language and meaning, is to follow. From this analysis it is possible to build a hierarchy of complexity, adding knowledge, which can only be expressed in terms of language and meaning, namely:

Knowledge
Information, language
Data
Files, records
Signs, signals
Computer representation, characters, words, bytes
Electronics (and now fibre optics?).

Following on from a statement made earlier in this section, at each level each of the following must be addressed:

Input and output
Storage
Processing
Transmission.

For these activities to occur in a controlled and reliable manner then

Control, i.e. management
Reliability and security

have also to be considered at each level. That is, at each level the discipline includes a study of what data/information is, and how it is handled.

Boundaries obviously occur with Philosophy and Sociology at the more complex levels, with Electronic Engineering at the more fundamental levels, and with, for example, Linguistics, Accounting and Statistics at other levels. It is suggested that this hierarchy describes the central core (see Fig. 1) of Computer Science and its many subdisciplines, and can be used to show how all these different areas of the discipline interface and integrate. And, in addition to satisfying the need for a unifying theme or philosophy, this approach provides the basis upon which to develop the more volatile elements to be studied, because it is largely independent of the state-of-the-art technology with an accepted body of theory that can be taught.

4. DEVELOPMENT FROM THE CENTRAL THEME

Such a proposition as outlined above can only be of use if it leads naturally into all the numerous branches of the discipline. To test this the comprehensive programme of the 5th Regulations of the BSc (Honours) in Computer Science of the Hatfield Polytechnic[17] was analysed using this approach. (Since this work was carried out the 6th Regulations of the degree have been introduced, starting September 1983, to replace progressively the 5th Regulations.) Fig. 2 is a diagrammatic representation of the scheme taken from the scheme description.

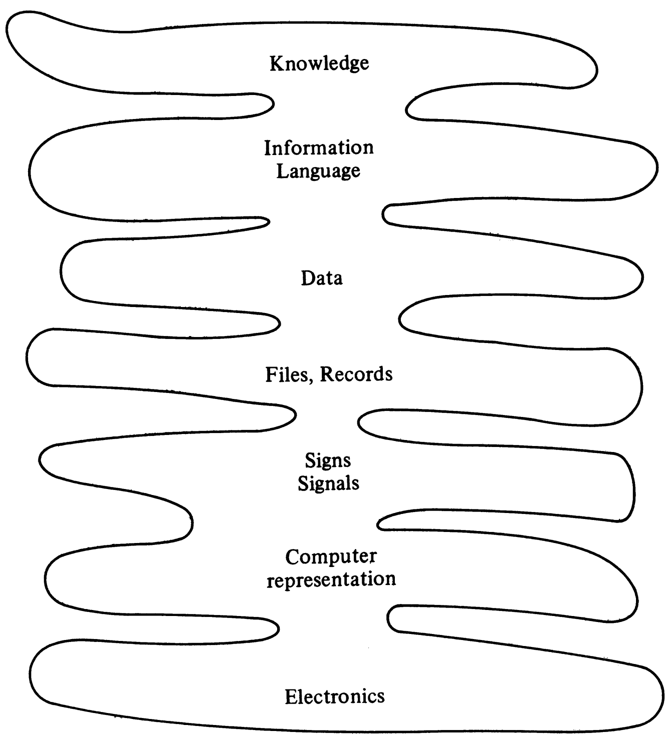

Figure 1. The central core of Computer Science.

A first analysis using Checkland's General Systems Theory approach and methodology[15] resulted in Fig. 3.

It is possible to show how the different elements of Computer Science with their own paradigms and algorithms lead from the central core of Data/Information/Knowledge. For example, how CSO (Computer Systems Organisation, a study of the computer and operating system) is a particular area of study with its own properties, algorithms and problems, dependent upon a study and knowledge of electronics and the manner in which data can be represented and handled. These properties, algorithms and problems appear only at these levels of Computer Science and can only be explained and understood at the lower level of Electronic Engineering. Equally synthetic languages are shown to have a similar relationship with a study of language, information (to be provided by the programs written in the language) and computer systems.

A more detailed analysis of the scheme of studies, but this time incorporating areas of study from supporting disciplines in the Social Sciences and Analytical Methods, resulted in the development of Fig. 4.

It must also be noted that:

- the size of the bubbles is a reflection of the size of the words and does *not* reflect in any way the importance or 'size' of the topic;
- the arrows indicate how the study of one or more topics can lead to and support the study of another topic at a higher level of complexity or by application of the material;
- the positions of the bubbles, apart from a general aim to place the more fundamental studies at the bottom and the more complex from which they were developed at the top of the page, were dictated largely by the needs of the drawing;
- where a single bubble contains two or more bubbles, the implied hierarchy also indicates that an additional degree of cohesion and integration between the sub-subjects is required;
- the analysis needs to be extended with some of the topics, e.g. Synthetic Languages, being further subdivided, but the writer feels that perhaps this is better left to the specialists in the areas concerned.

5. IMPACT OF THE ANALYSIS

It became apparent during this analysis that the use of this approach would impose a different structure. For example:

- support software, to include all aspects of software designed to aid the computer user, as one or one set of integrated course modules; traditionally such areas as graphics, mathematics, information systems and programming are covered almost independently;
- SAD, to include the analysis and design methodologies and principles applied to a wide range of problems and not to technical and commercial systems independently; this traditional separation is still probably the norm;
- synthetic languages; that the subject of programming languages, their need, design and implementation should be integrated.

Comparison of Figs 2 and 4 also indicates that the impact could go beyond this, and how the approach advocated in this paper can produce a very different structure for the total programme of studies. Accepting that Fig. 2 shows individual courses which Fig. 4 does not and cannot, it not being the representation of a specific programme of studies, the almost independent streams, it appears, would be replaced by a more tightly knit and integrated programme. But this need not prevent the running of modular programmes, options and streams of courses specialising in particular areas.

Fig. 4 suggests that optional streams allowing specialisation in such areas as Computer Systems and Microprocessors, Systems Engineering and Application Systems could be developed. Emphasis on one particular area, at the expense of others, could produce a degree in, for example, Information Systems.

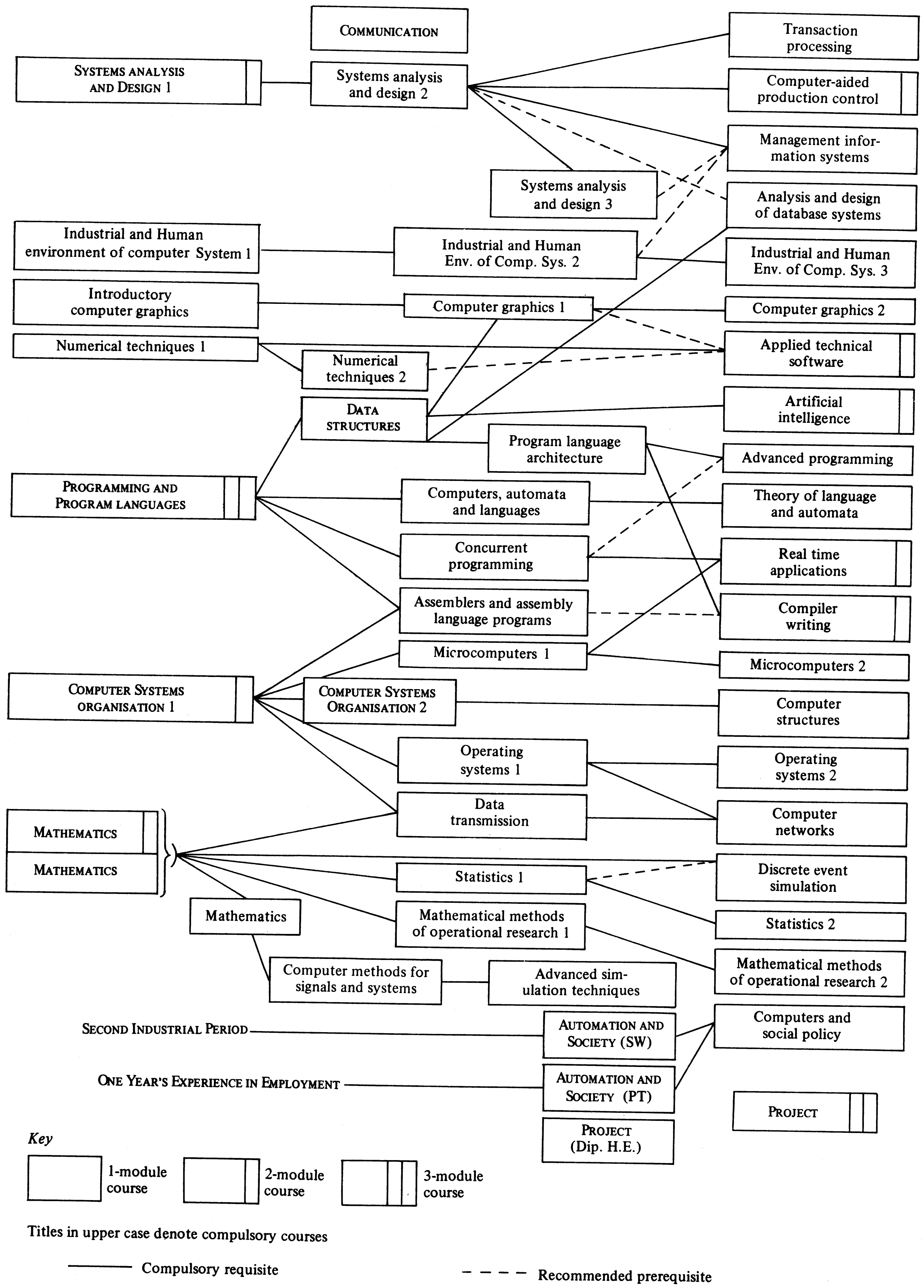
COMMUNICATION
SYSTEMS ANALYSIS AND DESIGN 1
Systems analysis and design 2
Systems analysis and design 3
Transaction processing
Computer-aided production control
Management information systems
Analysis and design of database systems
Industrial and Human environment of computer System 1
Industrial and Human Env. of Comp. Sys. 2
Industrial and Human Env. of Comp. Sys. 3
Introductory computer graphics
Computer graphics 1
Computer graphics 2
Numerical techniques 1
Numerical techniques 2
Applied technical software
DATA STRUCTURES
Artificial intelligence
Program language architecture
Advanced programming
PROGRAMMING AND PROGRAM LANGUAGES
Computers, automata and languages
Theory of language and automata
Concurrent programming
Real time applications
Assemblers and assembly language programs
Compiler writing
Microcomputers 1
Microcomputers 2
COMPUTER SYSTEMS ORGANISATION 1
COMPUTER SYSTEMS ORGANISATION 2
Computer structures
Operating systems 1
Operating systems 2
Data transmission
Computer networks
MATHEMATICS
MATHEMATICS
Discrete event simulation
Statistics 1
Statistics 2
Mathematics
Mathematical methods of operational research 1
Mathematical methods of operational research 2
Computer methods for signals and systems
Advanced simulation techniques
SECOND INDUSTRIAL PERIOD
AUTOMATION AND SOCIETY (SW)
Computers and social policy
ONE YEAR'S EXPERIENCE IN EMPLOYMENT
AUTOMATION AND SOCIETY (PT)
PROJECT
PROJECT (Dip. H.E.)
Key
1-module course
2-module course
3-module course
Titles in upper case denote compulsory courses
Compulsory requisite
Recommended prerequisite

Figure 2. Hatfield Polytechnic B.Sc in Computer Science, 5th Regulations.

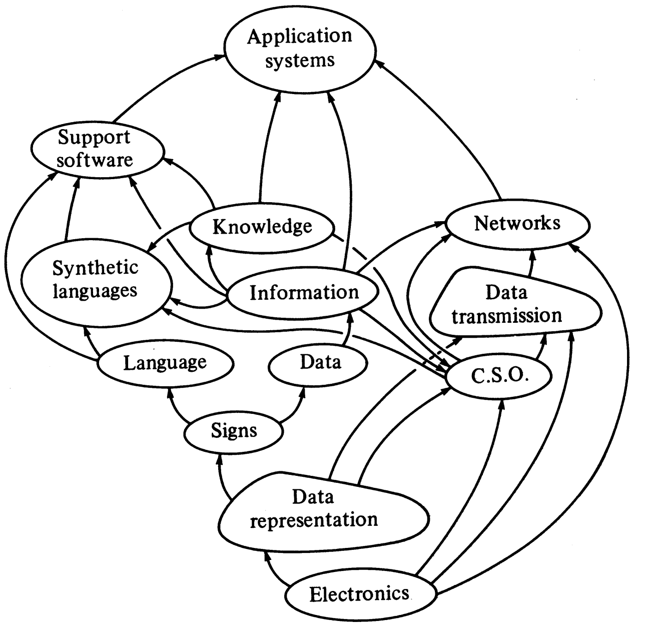

Figure 3. A simplified view of the structure of Computer Science.

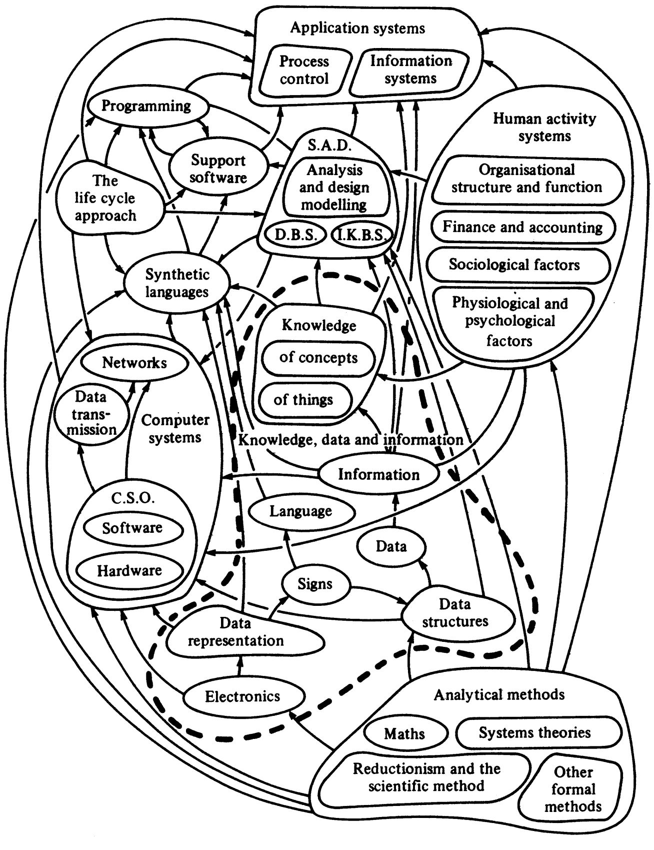

Figure 4. The structure of Computer Science.

6. A PHILOSOPHY FOR TEACHING

Computer Science is a technology and as such is concerned with the building of artefacts. Therefore, in simple terms, the prime aim of a programme of studies in Computer Science is to educate and train people to design and implement computer hardware, software and problem-orientated computer-based systems. Earlier parts of this paper have suggested that there is a cohesion and structure to the material, the subject matter, in the discipline that will provide an integrating force. But will such an approach provide a suitable vehicle for potential designers? These people are not primarily concerned with learning and understanding and possibly extending the subject matter, but in using it in a particular type of environment to solve or, more likely, to alleviate problems that other people encounter.

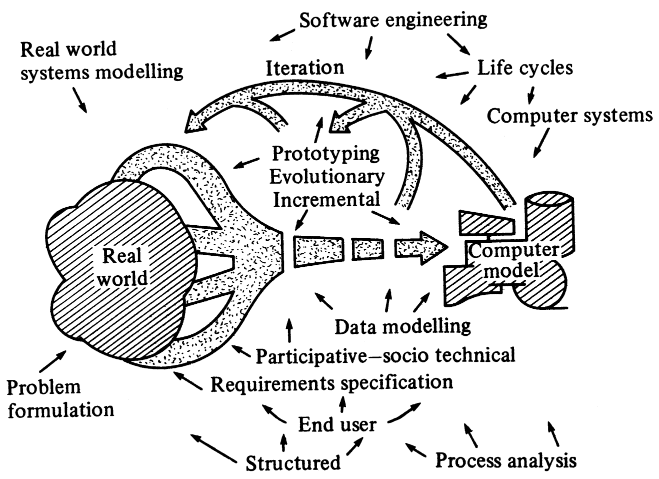

Figure 5. The three-part model and a multi-view perspective.

Given that all man-made products have to pass through some form of life cycle, and a systems or logistic approach is now widely accepted in engineering, then possibly a similar or software engineering view could provide a suitable theme for Computer Science education.

6.1 A software engineering approach

Jones *et al.*[18] described the need to build a three-part model of software engineering (see Fig. 5), namely:

- the real world into which there is a need to introduce some computer-based system (and hence which needs to be modelled),
- the computing system that is developed, and
- the transformation process of developing the latter from the former.

A number of issues arise from the use of this three-part model, namely:

- how can an existing object system, a human activity or socio-technical system, and its existing information system be understood and specified, and the problem understood and specified within its context?
- how can the requirements be analysed and specified in the context in which they will occur?
- how can the transformation be performed, managed and its correctness assured?
- how can the computer-based system be operated at an acceptable level of reliability and efficiency?

From these issues and the requirements they create it is possible to develop a number of separate courses, but courses that are linked to this central theme of analysis and design and covering a number of interrelated topics: i.e.,

- modelling of the real world,
- the technical stages in the transformation and proving process,
- managing the transformation and proving process,

- ensuring the involvement and support of the potential users, primary and secondary, of a new system,
- proving and managing the operation of a system, and
- formal methods and other subjects to support these topics.

But although this approach initially examined a narrow range of tasks, essentially the programming, it has here been extended to the upstream stages of information systems a d process control systems analysis and design, which are essentially part of the same process.

If the philosophical theme described above in section 3 is accepted, then the software engineering approach is applicable elsewhere, to the development of hardware, operating systems and languages, for example: they are essentially at a lower level of complexity of the same modelling process and are artefacts that have to be designed to meet requirements and then built to specification.

Unfortunately, although such an engineering-orientated philosophy may provide the approach to the teaching of material concerned with analysis, design and implementation, there is a very little likelihood that it can be taught as a single stream of courses, let alone a single course. Almost certainly a number of parallel streams, 4–6 being the most common patterns, will be employed. In turn this creates the problem of showing that these parallel courses are describing essentially different views of the same subject, not an easy or simple matter as students do not naturally bridge across courses. But it does indicate a need for some earlier and preliminary programme that leads directly into the separate areas and hence provides a single integrating foundation. The basis for such a preliminary programme has been described above in Section 3.

Where the programme of studies is long or, as with an MSc in Computer Science, is for mature postgraduate part-time students and is in a comparatively narrow and more naturally integrated area of study, then this problem will be much less evident. But, conversely, short (for example one-year) programmes for recent graduates will tend to highlight the students' difficulties because the programmes are so short and allow limited assimilation time.

To further reinforce, then, this process of integration, and to give the students some experience of using their recently acquired knowledge, it is necessary to allow them to complete some real individual project: to learn to design one must design.

6.2 Three-tier structure

Based upon the arguments developed in Section 6.1 it appears that a three-tier structure is required for a programme of studies.

Initially there is a need to present the basic material of the subject, describing at the different levels of complexity how data, information, language and knowledge are handled by the hardware and software. But this is only the essential core of Computer Science (see Fig. 4). For this material to be fully explored and understood additional supporting subjects, for example, Mathematics and Formal Methods, Organisational Structure and Functions and Social and Psychological Studies will be required. With a further course, in Systems Theory and Practice with its philosophy of levels and integrated elements, the whole foundation programme will provide a single and integrating platform for the second tier.

The second tier, concentrating upon the development of the students' analytical and creative abilities, should build upon the essential earlier foundation programme and be primarily concerned with the theme of:
- analysis of requirements within a dynamic and inter-relating environment;
- design and implementation of artefacts and the management of this process;
- management of the artefacts when in use.

Based upon and developed from this second-tier programme students must then carry out a third-tier project. This project should give them the opportunity to:
- apply the knowledge they have acquired to a practical problem;
- integrate further the taught material;
- develop their analytical and creative abilities;
- extend by self-study their knowledge in a required direction;
- communicate their ideas to others;
- develop their confidence.

This project should then complete the educational and training process aimed at instilling the basic core of the discipline and developing the analysis and design capability.

How each institute develops any programme and presents it will depend upon its own house style and the individual members of staff. Practical problems do have to be overcome, and the approach described in this paper cannot be applied precisely as described. It needs adapting to the needs and views of individual centres, and finally produced as a set of individual courses to be delivered within a limited period of time. It has also to be assessed.

7. CONCLUSIONS

The aim of this paper is to describe the author's own views of Computer Science (or it may be called Information Technology) and from that description to present in broad terms an approach to the development of programmes of study. A study of a particular area, namely Data, Information and Knowledge, can provide a central unifying theme for the earlier core material, an engineering-based and a problem-orientated design-driven approach providing a similar purpose for the later periods of study. Despite some emphasis being placed upon a 4-year undergraduate sandwich programme, it does appear suitable for a 3-year programme and a highly specialised postgraduate master's conversion scheme, in particular where the schedule is lengthened by preliminary courses and there is a need to reorientate and re-educate the participants. How each institute implements any programme will obviously depend upon the house style and the staff's specialities.

In the final analysis the programme will consist of a suite of courses taught by individual lecturers, within the constraints imposed by the time and resources available.

Acknowledgements

The author would like to thank his colleague, Mr M. Bacon, and the Hatfield Polytechnic for permission to reproduce Fig. 2, and the other members of Sub-group

B of the Information Systems Working Group, as some of this paper is based upon their discussions. The other members were: Brian Aspinal (Lancaster Polytechnic), Lesley Beddie (Napier College), Brian Bridge (North East London Polytechnic), David Chamberlain (Bristol Polytechnic), Ed. James (Imperial College), Jim Wood (Brunel University).

Equally of relevance and importance is the work of the author's colleagues, John Jones, Martin Loomes, Richard Mitchell, Wilf Nichols and Hugh Robinson, on the development of postgraduate software engineering programmes, and the author's gratitude must also be expressed to them.

REFERENCES

1. J. F. Jones and M. Loomes, *A Discussion on academic Strategies for the Development of Programmes of Study*. The Hatfield Polytechnic (October 1982).
2. *Principles and Regulations for the Award of the Council's First Degree and Diploma in Higher Education*, CNAA (1979).
3. C. A. R. Hoare, Professionalism. *Computer Bulletin* p. 2. (September 1981).
4. A Report of the ACM curriculum committee on computer education for management (editor R. L. Ashehurst) *Communications of the ACM* 363 (May 1972) **15** (5).
5. Curriculum recommendation for undergraduate program in information systems. *Communications of the ACM* **16** (2), 727 (December 1973).
6. J. F. Nunemaker, J. D. Cougar and G. B. Davis, Information systems curriculum, recommendations for the 80's. *Communications of the ACM* **25** (11), 781 (November 1982).
7. J. N. G. Britton (Ed), *An International Curriculum for Information Systems Designers* (IFIP Committee TC3), I.B.I.C.C. (1974).
8. Draft report of the revision of the IFIP/ISD Curriculum (September 1982).
9. Report on degrees in data processing (BCS working party, editor R. Shaw). *Computer Journal* **18** (4), 382 (November 1975).
10. L. Capper, *A Philosophy for Computer Science and Information Systems* (Report of IFIP WG8.2 Meeting of December 1982).
11. L. Capper, Information Technology and the BCS, *Computer Bulletin*, p. 14 (September 1983).
12. L. Capper, *ISWG, Report of Sub-group B* (March 1982).
13. L. Capper, *ISWG, Report of Sub-group B* (April 1982).
14. L. Capper, *ISWG, Report of Sub-group B* (July 1982).
15. P. Checkland, *Systems Thinking, Systems Practice*. Chichester, Wiley (1981).
16. R. Stamper, *Information in Business and Administrative Systems*. Batsford (1973).
17. *Course Description of the 5th Regulations of the BSc and DipHE in Computer Science*. The Hatfield Polytechnic (June 1980).
18. *Scheme Description, Postgraduate Diploma in Software Principles and Practice*. The Hatfield Polytechnic (December 1982).