WORKING NOTE AC73

Alan Creak
15 November 1989

COMPUTING - A COURSE STRUCTURE.

ABSTRACT

(- supplied because no one would actually read as far as the interesting bits if I put them any later.)

I suggest that the primary organisational principle of our presentation of computing to students should be a sequence of well defined and clearly articulated aims, one for each stage of the overall course. My recommended set of aims is :

Stage 1 : Understanding simple computing. What happens when we use a computer ?

Stage 2 : Understanding (currently conventional) computing in general. How does it all fit together ?

Stage 3 : Making computers work for people. How the principles are applied today.

Stage 4 : Preparing for the future. Present research and future developments.

This primary structure should be extended by defining a set of topics, chosen to give adequate coverage of the material of the course, each of which should be dealt with in each year's presentation. This is not a recommendation that specific lecture courses be provided for each topic in each year, but a device for identifying material which should be there somewhere. There are several ways in which the sets of topics could be chosen. One example is :

Hardware : how computing machinery works, and why.

Software : how the hardware can be constrained to work in useful ways.

Data : organising, structuring, storing, and moving the working substance.

People : how we can communicate with, control, and coexist with computing activities.

MOTIVATION.

This note is an account of my view on the design of a course structure for a university computing department. I emphasise that it is a personal view, and I make no attempt to compare it with any other course proposals. My object in writing down the material is to set out the reasoning behind my views so that I can look at it critically and satisfy myself as to its coherence.

In the process, I am sure that I shall find points at which the views I hold at this moment need amendment, so the result will almost certainly differ from my present ideas in some respects. By the same token, I shall not stop thinking when I complete the report, and my views will continue to change; so this account should be seen as an indication of my position now, as an invitation for comment and criticism, and as no more (and no less) than that.

Note added afterwards : in writing this report, and particularly in trying to follow the programme outlined in the first paragraph, I find that I have produced a notably uneven result. One very obvious feature is that I have spelt out some parts in much greater detail than others, because it seemed important at the time to do so. Sometimes these passages mark bits I hadn't thought of before, and sometimes they mark changes of opinion - for the second paragraph was indeed borne out in the writing, and the exercise has enlarged my view in many respects.

WHY DO WE NEED TO DEFINE STRUCTURE ?

- to which one response is, "Why do we need to ask 'Why do we need to define structure ?' ? Surely this is mere preciousness, pseudoacademic pseudophilosophical posturing !" I would probably think the same myself were I to find this heading in someone else's report; but I include it nevertheless because I found it helpful to address the question quite specifically when working out my own ideas. Perhaps because computing is still an immature field, it isn't always obvious how individual topics fit into the whole. There are rather few guidelines which can be used to assess the significance of any potential component of a course.

It is interesting to compare the position of the physical sciences, where there are clearly defined deep principles - atomic theory, electromagnetism, relativity, and so on - from which many other topics can be developed. Similarly, in the traditional engineering disciplines, everything except the raw material is built from something, and the structure imposed by such constitutional factors is useful in organising the subject. Mathematics offers yet another pattern, in which order is imposed by the strict development of topics from other topics and, ultimately, from an axiomatic base.

Now, it is important to notice that even when such structure is discernible in a subject, it is not necessarily directly reflected in the organisation of university courses. The first stage 1 chemistry lecture of the first term does not necessarily begin with Schrödinger's equation, and similar remarks apply to the other disciplines mentioned. Of course, the underlying structure is an important part - arguably, the most important part - of the material to be presented in the course; but it is not necessarily practicable to use it as a guide to organising the course itself.

In fact, the reason for defining structure in a course is not to mirror the logical structure of the subject matter : it is rather to promote effective learning. In practice, this seems to imply that material be grouped into collections within which many common principles apply. So, chemistry is divided into inorganic, organic, and physical chemistry (sometimes augmented by theoretical chemistry or analytical chemistry, if that suits the whim of the local establishment); then inorganic chemistry is divided into typical elements, main groups, and transition elements; and so on for as long as it seems convenient. Of course, the divisions are artificial, but they are not arbitrary : there is a collection of ideas within each defined field which, because of the nature of the topics studied, are less useful outside the field. Artificial or not, the divisions serve very well as devices for introducing the subject in early years of the course. Later, there are holes to fill in : topics like organometallic chemistry, physical organic chemistry, and spectroscopic methods help to bridge gaps.

A second reason for defining the structure of our course is purely utilitarian : for effective course planning. It is not practicable to present courses amorphously; lecturers have to specialise to some extent, students have to know what topics are treated when and where. For students wishing to learn about specific topics rather than computing as a whole it is important that material be grouped into coherent and useful chunks.

For these reasons, structure, even if not evident in the subject itself, must be imposed : and I believe that the considerations I have outlined in the preceding paragraphs suggest ways in which this can be accomplished. It is true that, whatever the means adopted, boundaries between topics may appear to be contrived. If the analogy with other subjects is a good guide, this may not be very important - provided that the artificiality is understood, that the groupings used do collect together material with a strong internal consistency, and that some way of filling in the gaps is provided in more advanced coverage.

When I started this exercise, I thought of computing as in some ways a flat subject, characterised by many possible topics for treatment but with rather little depth of intrinsic structure. (In previous debates on related topics, I have been taken to task for similar assertions, and topics such as information theory and computability theory urged upon me as the roots from which computing grows. If that's so, then the computing tree has a very long and practically invisible trunk, much like that joining the chemistry of paper to literary criticism.) Now I've thought a bit more, I'm less certain about that. I have found enough structure in the central topics of computing to make sense of a three-year degree course (though it doesn't necessarily spring from the deep roots recommended by my earlier critics); but now I suspect that the overall picture has been blurred by including within the boundaries of the subject some topics which perhaps should not be there. These are certain topics which depend strongly upon computing techniques, and are very hard to develop without using computers. Examples are artificial intelligence, robotics, computer graphics; I shall mention them again later.

WHAT IS TO BE DEFINED ?

Planning the course we offer comes down to answering two questions : what should we do ? and when should we do it ? We must select topics for treatment, and decide in which order to treat them. An implication of the need to select topics is the need to reject topics : a decision to include topic A may also be, in effect, a decision to exclude topic B, simply because the time available for the course is limited. How should we go about making this selection ?

In traditional areas, we can use the underlying structure of the subject to sort out what should be presented. We can make some sort of judgment on the importance of a topic by identifying its position in the hierarchy. Even then, though, structure alone may not be sufficient : it can sort out basic - and presumably essential - material from peripheral matters, but cannot decide between topics on the same level. Does that matter ? That depends on whether or not the problem will arise.

If my original assessment of computing as a flat subject is correct, it has in any case comparatively little in the way of clear structure, and is largely composed of topics on the same level. To set these into some useful order, we would have to appeal to some other criterion; this is where we can introduce our aims in presenting the course. The effect of such course engineering is to impart a bias to the course. (This is a perfectly legitimate thing to do, and we can make it sound much more respectable if we speak of tuning the course to match our perceptions of the important areas, etc. etc.) We would end up with a course emphasising software engineering, or commercial aspects, or computing theory, or whatever.

If, on the other hand, the core material of computing is sufficiently well structured, we may not need to make these decisions. I now think this is more likely to be the case. We may still end up with a biased course, but that will be because we've had to decide what to cover at stage 3, rather than because we have too many little topics to cope with. In effect, we shall be drastically pruning a tree rather than trying to tie together a clump of bushes.

I shall assume without further debate (on my part, anyway) that we intend to present a respectable academic treatment of the topic, by which I mean something like a fair survey at an appropriate level of the field under discussion, within which the importance of any component can't be judged except by internal criteria (typically related to the breadth of application of the component). That is not to preclude assessment of topics by other criteria, which may well itself be legitimate material for discussion : so we may include the bubble sort if it illustrates important principles, but also remark on its expense when used with large collections of data. It *is* to preclude deliberate omission of topics because they're inconvenient, or show IBM in a favourable light. The implication of this decision is that we must provide proper discussion of topics we cover in the courses. Any appropriate background material must be presented somewhere, and links between different parts of the subject must be made as required. In other words, we cannot offer a course on, say, very large databases without making sure that necessary supporting material on database theory, multiprocessing systems, or whatever other background may be needed is either treated in the course or covered in related courses.

What room for manœuvre is left ? If we read the previous paragraph as committing us to providing proper background material for all our courses, then the question is almost equivalent to asking what topics we should treat at stage 3, and to what depth, and then working out how much time remains unused. (The "almost" allows for the possibility that the subject is so flat that there is a dearth of topics which cannot be taught until after stage 2; as I've already remarked, I do not believe that this is the case, and will not pursue this possibility further.)

Finally, we may ask a question concerning the treatment of the material. Following the three patterns I mentioned earlier, should the approach resemble that proper to science, engineering, or mathematics ? (Other possibilities - commercial, fine arts - preclude a fair survey of the material.) In my view, the engineering pattern is most appropriate; but it probably doesn't matter much which we choose, provided that we stick to the principle of fair treatment. In the descriptions below, I shall give suggestions for all three possibilities, but I regard these more as pointers to the way I see the topics than as serious proposals.

In summary, I believe we should make a real attempt to organise the material of the course in a two-dimensional space, with axes for time and topic. For each part of the course, we should be able to say clearly *when* it should be presented, and *what* topic it concerns. I choose to deal with the "when" question before the "what", because I find that more constructive - knowing something about the sequence of events helps to determine the context of the events, and therefore what material can sensibly be associated with the different periods. The alternative route gives little guidance on matters regarding the quantity of material required, and ends with a packing problem.

THE TIME DIMENSION.

While there are certain sequences in the material which it would be foolish not to observe, the order of presentation is not in general fully determined by selecting the topics to be covered in the subject. For example, one could choose to cover "important" topics fully in the first year or two, leaving less significant material until later; or one could choose a "spiral" approach, giving more general coverage each year but advancing the depth of treatment year by year.

A sensible solution to this problem of course organisation could start with the question : how do people want to use the course ? If it is intended primarily as a specialist course in computing, then we may choose whatever approach is convenient for the course organisation as a whole; but if it is also available to non-specialists who may just want to attend all or part of our course for one or two years, a spiral approach would probably be better. As our lectures are in principle available to anyone in the university, subject to such constraints as prerequisites and faculty regulations, I have chosen to use the spiral approach, taking note, of course, of any essential sequence constraints imposed by the natures of the topics themselves.

The question now becomes : how far should we take each topic during each year of a three-year degree course ? That isn't an original question, as the spiral approach I have recommended is broadly the same as the traditional approach we already use. I think the answer had better be different, though : so far as I can tell, such decisions are now taken without any clear guidelines which pertain to the course as a whole. The standard of argument seems to be "course 07.3nn needs a knowledge of thing Z, so we'd better put Z into 07.2mm, which will make 2mm too full, so we'd better move thing Y to 2kk ....".

The answer I propose, ex hypothesi better than our current practice, is to set overall goals for each year of the course. We cannot sensibly set goals independently for each individual lecture course - except those we can satisfy locally, such as self-containedness - because they must be constrained by many external influences, such as the other courses. We can, and I think we should, set such goals for the computing course as a whole. The goals should be simply expressible without undue recourse to technical terms, so that we can tell prospective students "if you follow our full course up to stage N, you should be able to do X".

What difference does it make ? I think that the main effect is to make it easier to decide at what level various subjects should be covered. If material Q is thought to be needed to meet the stage 2 goal, then either it must be included at stage 1 or stage 2, or we must decide that we don't need Q after all. That illustrates a phenomenon which I guess would be a significant consequence of accepting this structure : we will be constrained much more forcefully than at present to evaluate the material of the courses, and to reject inessential components. It is worth making the point that such evaluation is not easy, and takes a lot of time, both in itself and in collecting and preparing any new material which is to replace the old. The aims I suggest below culminate, at the undergraduate level, in a presentation of the current state of computing : in a subject which changes as rapidly as computing, any serious attempt to keep up to date over a broad front should be recognised as a significant component of an academic's work.

Is it possible ? I think it is, and the suggestions I offer below constitute my attempt at an existence proof. I don't remember seeing anything quite like it anywhere else, though. (It's true that I haven't looked very hard.) There was certainly no feeling of moving through a subject by stages in my own undergraduate work - each year was just rather more of what happened last year, which I suppose means that it was an archetypal example of the spiral approach. Of course, new areas were introduced (usually) after the material on which they depended had been covered : and, contrary to my proposed approach, every new topic, at whatever level, seemed to start off in much the same way.

Here, then, is my recommended set of aims. I've tried to describe each step in the spirit of each of the three emphases I mentioned - science, engineering, and mathematics - to suggest different approaches. The results are less than totally convincing, and should not be taken too seriously, but illustrate the idea.

Stage 1 :	**Science :**	how does a computer work ?
	Engineering :	what happens when we use computers ?
	Mathematics :	defining terms.
	Simple English :	what happens when we use a computer ?
Stage 2 :	**Science :**	filling in the gaps.
	Engineering :	are there any better ways ?
	Mathematics :	exploring the properties of the fundamental items.
	Simple English :	how does it all fit together ?
Stage 3 :	**Science :**	how the separate topics interact in practice.
	Engineering :	making computers work for people.
	Mathematics :	applying principles to problems.
	Simple English :	how computer systems work today.
Stage 4 :	**Science :**	how will computing develop ?
	Engineering :	how can we do it better ?
	Mathematics :	extrapolation.
	Simple English :	present research and future developments.

THE TOPIC DIMENSION.

In deciding what topics should be covered in a university computing course, we have to answer several questions. The answers should define what material must be presented in our lecture courses, and also tell us enough about the organisation of the material to show us how we can usefully distribute it between a number of separate courses.

We begin with what we might call the zeroth law of computing : there is far too much material. The first task is clearly to set some sort of boundaries which we can use to determine whether or not a candidate topic should be considered as material to be included in the course. I think there are two questions here : first, how do we distinguish "real" computing from applications ? and, second, how do we distinguish essential material from inessential ?

I shall not address the second question directly, because I have no good answer. In practice, it will probably answer itself : we'll select a plausible set of stage 3 courses, and everything else will perforce fall into place. But I'd like to know what it is that makes the stage 3 set look plausible.

On the first of the questions, I must hasten to add a word of explanation lest I be dubbed elitist, which would be a dreadful fate and would naturally invalidate anything I had to say. A payroll programme (or a matrix inverter, or an arcade game) may be an excellent example of some computing technique, and can legitimately be studied for that purpose in a computing course; but our interest in that technique is not directly connected with the payrollness of the programme in which we find it. Indeed, if it only appears in payroll programmes, it is probably not something which we should treat. Our concern is with the general principles of computing, whatever they may be. (Notice, by the way, the effect of replacing "payroll" in the foregoing sentences by "artificial intelligence", "computer graphics", or "robotics" - or, doubtless, other areas, but these come to mind. There is, at the least, a question to be answered if we wish to include these topics in our course.)

So how can we identify these "general principles" ? One criterion suggested by the preceding paragraph is that of ubiquity : if it turns up in a lot of contexts, it's general. A second criterion is that of permanence; it would clearly be nice to build our subject on foundations which won't change every time someone releases a new operating system. Others are doubtless possible, and their elaboration is left as an exercise for the reader. I shall not attempt to identify the general principles, because that isn't the object of this discussion; instead, I want to focus on the criteria, because the criteria which identify the principles of the subject must presumably be related in some sense to the topics we are seeking. More precisely, since the topics are something like a set of headings under which we can collect the principles when we find them, the criteria for the principles must characterise the topics too. We can reasonably add a criterion specific to the set of topics : that it must not be too large. It is a good principle of top-down analysis that we do not decompose an entity into a large number of different components; three or four seems to be a reasonable number in practice, with the magical seven (± 2) as some sort of upper limit.

It is interesting to compare the traditional hard sciences of chemistry and physics. I have already remarked on the traditional structure of chemistry; similarly, physics is (or certainly used to be) seen as a composition of heat, light, sound, propertiesofmatter, and magnetismandelectricity. What classification can we discern within computing which might work in the same sort of way as these illusory but effective dividing lines ? There are several possibilities, of which I shall mention two.

A candidate based on a fairly traditional view of computing is the set :

hardware; software; data; and people.

Of these, the first three are traditional, and need no comment. I have included "people" as an additional topic because I believe that, though the need to communicate with people has been a neglected factor in the development of computing, it is in fact the reason why there is a subject at all, and is likely to become increasingly prominent as a natural consequence of the emphasis on bringing computing technology to more and more people. It is, indeed, arguable that with the very high rate of change in computing technology, one of the few (fairly) reliable invariants we have is the needs of the people who use the machinery. Obviously there are lots of interactions between these topics. I don't see that as particularly important - and it may even be an advantage in demonstrating the artificiality of the divisions. The important feature is that the classification identifies a useful set of viewpoints. Some may prefer to change the set of topics; not everyone will agree with my arguments for including "people", and an obvious contender for inclusion is the theory of computing. In each such case, a decision is necessary : would it be better to include appropriate elements of the theory in the material of the various courses when it becomes appropriate - or is it better to separate out the material into a distinct stream for convenience ? In making decisions of this sort, it should be borne in mind that to have too many topics would impair the utility of the classification.

What about learning a programming language ? I accept that this is something of a vexed question, but I do not believe that it should be a significant feature of the formal material presented in any course. That does not mean that programming language constructs should not *appear* in any course; it means rather that courses requiring programming constructs which have not been used before must take care to introduce them, in context at the appropriate time. Details can, and should, be gleaned from textbooks.

An alternative classification can be based on how we use computers rather than on the natures of their parts - perhaps an engineering rather than a scientific criterion. Here is a sample trichotomy :

What do we want done ? (data, their relationships and operations thereon, storage and retrieval);

Ways and means for doing it (hardware and software, structure and architecture for both);

How do we say what we want ? (interfaces, languages, data presentation).

I should perhaps remark that my motives for introducing this scheme are not merely to persuade you that it's the right one, or the best one (though it does have a certain appeal). I present it principally to show that other approaches are possible. Doubtless this is only one of many; but any such new view carries the considerable benefit of a new and relatively unpreconditioned assessment of the material to be presented in the course. For example, the last category in the list above groups together the various topics connected with how people instruct machines to work, and includes traditional programming and job control languages, graphical and spreadsheet models, natural language techniques, and so on. That sounds to me much more fun than the traditional categories, where topics do rather tend to stay in their boxes. (It also sounds like much more work and disruption, but that isn't the point at issue.)

FITTING THE TOPICS TO THE TIMES.

Having identified coordinates along our two dimensions of time and topic, we must now try to draw them together into a coherent course in computing. The first point to make is that we needn't necessarily just draw the obvious matrix, fill in the topic names, and leave it at that. In other words, a decision to include certain topics at certain stages in the overall course doesn't mean that we have to say so. The names of the courses need not necessarily be identical to the names of the topics, and we need not provide specific lecture courses to cover individual topics. Our specifications of material to be covered at particular stages in the course are only a checklist which we can use to classify the available material and to identify omissions. So far as the actual courses in the Calendar are concerned, it may be preferable to collect together the material in ways depending on, say, application areas. Other things being equal, we might perhaps expect that we would wish to present the "pure" topics first (following the classical scientific pattern of abtraction), but that interactions between the topics would become more prominent in later years; once again we can find precedents in hard scientific disciplines, as in polymer chemistry or solid-state physics.

I offer the remarks which follow simply as a suggestion of considerations which might be thought important in determining what sort of courses to present, and as a stimulus to discussion. I do not put forward specific proposals for a syllabus.

Stage 1 needs a fairly amorphous general course, if only because students can't be assumed to understand the terminology yet. One of the jobs is to introduce the distinctions between the topics - and, at the same time, to point out relationships between the topics. Each of the topics would therefore be represented at an introductory level in a single course, called something like Introductory Computing.

Stage 2 students can be assumed to have absorbed the general structure of the subject well enough to understand a bit about the interactions between the parts without having it spelt out all the time, so now we can specialise a bit. (But not to excess !) Some sort of specialisation is probably necessary anyway, because this is where students begin to pick and choose - for example, electronicists might only be interested in computer hardware.

Stage 3 brings a choice. We can stick with the stage 2 topics (easy for us, for prerequisites, and a traditional pattern for some science subjects); or we can start new groupings, perhaps by fields of application (nice and coherent, perhaps, and closer to our current practice). Subjects like operating systems, which depend on more than one strand of the stage 2 course, fit in here.

Stage 4 is, in practice, likely to be dominated by lecturers' interests, so it is probably unhelpful to worry about details of organisation.

In reading this list, bear in mind the goals for the various stages as suggested on page 5. While there are perhaps many ways to make it all work, my expectation (as I've hinted a few times already) may be worth presenting as an example. I would expect a general first year. The second year of lecture courses would be devoted roughly to the "topics" in purish forms, in which the stage 1 introductions and definitions are broadened, elaborated, and generally explored. The third year would consist of courses more directed towards groupings which reflect the practical application of the earlier work : machine architecture, networks and data communications, databases, operating systems, communicating with computers, software design and implementation.

CONCLUSION.

There isn't one, really - except that I think my suggestions are workable and desirable, and I'd like everyone to agree. I do believe that the course we offer sahould be subjected to careful scrutiny : I don't think that at the moment we can easily say what we're trying to do in it, or why different topics are there. And I think my suggestions do it better.

TRENDY POSTSCRIPT : INFORMATION TECHNOLOGY.

I don't like the name "computer science" (see the next paragraph). If I were consistent, I'd prefer "information technology". I don't. The "information" bit is all right, though I don't think it's taken seriously; the "technology" is just as gratuitous as the "science" in "computer science", and just as nasty a means of trying to imply respectability by association, even if it is a shade more realistic. It is true that nobody has asked me for my views on the name, so I shall proceed to discuss the denotation.

Is it any different from computing ? Or is it more of the same, perhaps with a different emphasis ? My impression is that the term is of British origin (I don't seem to see it very much in the US publications I read), and was coined by some Thatcherite political hack to shift attention away from scientists (who waste time and money on fundamental research which doesn't show up in this year's balance sheet) and towards technologists (who, supposedly, keep their eye on the main chance and make a profit now). The hack in question clearly didn't know the technologists I do. I could go on, and often do.

Polemic apart, I suppose we're stuck with the term, and I think we have to interpret it as the mixture as before, perhaps with an emphasis on using technical means to do things better, to give better service, and maybe to do new things. If that view is correct, I think it's covered in the material I've already given.

THOUGHTFUL POSTSCRIPT : WHAT'S COMPUTING ANYWAY ?

One of my reasons for disliking the name of this department is its insistence that what we do must be tied to an arbitrarily selected machine called a computer. It is a historical accident that the first really effective means we have found (apart from ourselves) to carry out a certain class of interesting and useful manipulations on data (whatever they are) was the electronic digital computer; but to invert this observation by making the electronic digital computer the criterion of what's interesting and useful does not seem helpful. The façade is already cracking : neither neural networks nor robots are computers in the ordinary sense of the word, but both perform interesting and useful operations, of sufficient complexity and importance to be worthy of study in their own rights. Even computers seem to be turning into desktops - because the properties of desktops are deemed to be more useful than those of computers for getting work done.

The French originated (I think) a subject they call "l'Informatique" (which is surely highly acceptable to modern and trendy ears; it even hides its gender), sometimes translated into "Informatics" (which I think looks silly, but better than "computer science"). I have recently come across a plea for another name, which I have forgotten, derived from Greek roots. That it was proposed by a Greek was, I'm sure, coincidental. Whatever one's view of the name "<?>informat<?>", the intention seems to me to be laudable, as it focuses attention on the important part of the subject.

So far as our own courses are concerned, I think that this question about the nature of the subject is more than idle speculation. Clearly enough, at the moment it will make little difference, as digital computers remain the only means we have of doing the work we're interested in. With an eye on the future, though, perhaps we should not insist on restricting our treatment to conventional digital computers. Our students will be working in the future when it comes, and a little judicious foresight, even if it's wrong, could very helpfully broaden outlooks. How much of our courses would survive a transition to dataflow or reduction machines ? What can we do better, or worse, with neural networks or analogue computers ? What can we do at all with optical computers or robots ?

Two or three years ago, I circulated a mildly amusing (well, that's what I thought) challenge which amounted to a request for an optimising compiler for the U-Bix photocopier. It was only half in jest. The U-Bix accepts instructions in a less-than-trivial language, and interprets them to guide its subsequent actions - which can be quite subtle, particularly if something goes wrong. It certainly implements algorithms, which some have suggested as an important characteristic of "computer science". Since my challenge, the U-Bix has become even cleverer : true, it still falls a little short of being a programmable machine in the sense of a computer, but it's in the pocket calculator range. Is this the shape of things to come ? Do we care ?