Education Grand Challenge:

A New Kind of Liberal Education

Making People Want a Computing Education For Its Own Sake

Aaron Sloman

(http://www.cs.bham.ac.uk/˜axs/)

Introduction

Nearly thirty years ago (partly inspired by writings of John Holt, Ivan Illich and Seymour Papert) I had a vision of computing as the basis for a new kind of "liberal education" expressed thus in (Sloman 1978)

> *Another book on how computers are going to change our lives? Yes, but this is more about computing than about computers, and it is more about how our thoughts may be changed than about how housework and factory chores will be taken over ...*
>
> *Thoughts can be changed in many ways. The invention of painting and drawing permitted new thoughts in the processes of creating and interpreting pictures. The invention of speaking and writing also permitted profound extensions of our abilities to think and communicate. Computing is a bit like the invention of paper (a new medium of expression) and the invention of writing (new symbolisms to be embedded in the medium) combined. But the writing is more important than the paper....* (The Preface) ...
>
> *From early childhood onwards we all need to play with toys, be they bricks, dolls, construction kits, ... scientific theories, or other people. We need to interact with all these playthings and playmates in order to develop our understanding of ourselves and our environment that is, in order to develop our concepts, our thinking strategies, our means of expression and even our tastes, desires and aims in life. The fruitfulness of such play depends in part on how complex the toy and the processes it generates, and how rich the interaction between player and toy are.*
>
> *A modern digital computer is perhaps the most complex toy ever created by man. It can also be as richly interactive as a musical instrument. And it is certainly the most flexible ...* (Chapter 1)

What happened?

For most people a computer is just a tool, like a food-processor or an oven. You can use it to do things you previously wanted to do, as long as you are taught to use it. But that merely involves speeding up what could be done before, e.g. producing and transmitting pictures and text, or collecting pictures and text produced from others, for entertainment, scrap-books or school projects. And if the computer does not do what you want, you do without it or consider buying another one. If you are well enough informed you may download a new package from somewhere, like buying another gadget for a food-processor. You have no thought of re-programming it yourself: you are de-skilled.

There are also armies of drones who earn a living by making the computers do these things, mostly by adding small variations to what has already been done by a tiny group of visionaries who enhance the state of the art by designing new hardware, programming languages, operating systems, or types of applications.

Another tiny minority of professionals contains mathematicians fascinated by the problems of formalising with precision and elegance what can be thought about structures and processes which can occur in computers.

Last of all there is a miniscule subset for whom the 30-year dream has come true, either because they had visionary teachers or because they somehow discovered the mind-stretching powers of computers of their own accord, because the opportunity was there.

The challenge is to make it come true for the majority.

What does that mean?

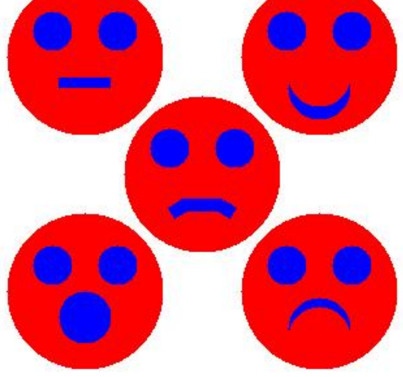
I am not talking about teaching more people to use web browsers, word processors or spreadsheets. I am not even talking about teaching them how to produce their own web pages.

Consider the difference between (a) using an editor or email program to compose some text, and (b) feeding some rules into a sentence generator to produce text. The first, (a), involves using linguistic competence that everyone has, coupled with a new device for linguistic expression, just as a pencil was once a new device, whereas (b) requires reflection on what one's own language is, how it works, and what constraints it conforms to. It requires making knowledge explicit that we all have implicitly.

However it is difficult to do and easy to get wrong. So even with powerful tools at hand one's initial efforts at getting the machine to print sensible sentences will produce some garbage, requiring a process of analysis and debugging, not of the program but of one's theory. This can happen at different levels of sophistication. E.g. even a child could learn to write 'Eliza' programs of the sort sketched here http://www.cs.bham.ac.uk/research/poplog/teach/respond .

At a more sophisticated level instead of a pattern-based reply-generator, one can expose learners to programs that generate sentences, or larger structures such as haikus or stories, on the basis of a user-supplied grammar and a lexicon, and give them the opportunity to produce a generator for story openings, or for railway station announcements, or for wedding announcements, etc. Experience with http://www.cs.bham.ac.uk/research/poplog/teach/grammar shows that learners (including girls who have no wish to learn programming) can tackle the task with enthusiasm and, as before, find that the results include garbage, which requires debugging of their theories.

Consider the 'faces' shown on the right, drawn in two colours. They could be drawn using crayons, or paints. But suppose a child is given two procedures, one for making a circle and one for making a bar, of a specified size, colour and location. I have watched non-mathematical learners copy and edit instructions for using such procedures, then discover, with some hints, and much debugging, how to produce the face on the bottom left using one big and three small circles, then how to add another circle to get the faces on the right, and then perhaps seeing how a bar-drawing procedure can produce the others. Even for adults doing this can produce delight. Some learners go on to generalise and parametrise face-makers of different sorts.



It also produces what for many is a new understanding of the structure of a 2-D surface, and the ways in which processes in a surface can be specified and how such processes interact. They learn mathematics they thought they were incompetent to learn.

Learners often enjoy the opportunity to produce behaviour, e.g. playing games or acting out scenarios themselves, e.g. playing make-believe, playing charades or performing in plays. Compare this with giving them tools to specify behaviours of 'toy' interacting individuals controlled (e.g. on a screen) by a computer. The individuals may interact purely textually e.g. in a simulated quarrel or a simulated meeting of two friends. In some cases the behaviours could use graphics, as in these demonstrations: http://www.cs.bham.ac.uk/research/poplog/figs/simagent

In all these cases the learner can choose goals then aim for them, and the goals will have a kind of clarity that makes it evident when they have not been achieved (e.g. the quarrel does not look like a quarrel), but in a context that encourages analysis and exploration to find out why not, and usually produces motivation to debug the theory and try again.

This principle is applicable to a *vast* variety of different tasks at many levels of sophistication and depth, some as simple as the old LOGO task of specifying movements for a mechanical turtle using a 'button' box, others as complex as designing an interactive game with many characters all behaving independently and autonomously: a wonderful collaborative design task.

In all these cases, besides making the machine achieve the selected goals the learner can also be put in a social situation where there is a need to communicate how the system works, what it is supposed to do, what is going wrong, how it might be fixed, why one strategy works better than another. For younger learners this could simply be embedded in classroom interactions. For older learners the analytical descriptions could be produced in writing with the aid of diagrams, etc. where appropriate. Self-critical project reports should be encouraged.

Conjecture

I conjecture that (a) there is a vast space of possible ways of using computers to provide learners of all ages with new, enjoyable, mind-stretching ways of making computers do things that the learner specifies, including simplified versions of things the learner can already do easily, (b) in the process many different things will be learnt about the subject matter of the task (e.g. human language, 2-D structures and processes, interacting agents, music, etc.), (c) in particular, as the tasks increase towards more human-like performance the learner will gain new insights into what it is to be human, (d) the requirement not only to produce results, but also to analyse and explain, will help to produce more analytical and articulate learners who become better at thinking about complex matters and communicating about them, (e) learners will become more and more accustomed to thinking about abstract processes in abstract (virtual) machines and be in a better position to use the understanding gained thereby in thinking about aspects of reality where there are structures and processes at different levels of abstraction, including structures and processes in brains and minds.

I do not claim, as many do, that the way to make progress is to design some new *programming language for beginners*, whether logical, diagrammatic, based on natural language, based on graphical gestures, etc. Instead different sorts of tasks will require different kinds of specification languages, as should be obvious from the examples of linguistic, graphical, musical, and agent behaviour tasks.

The challenge

The challenge is to analyse many task domains that are amenable for treatment in the manner outlined above, for learners of all ages (obviously building on work already done by gifted teachers with and without computing skills), and then to devise a collection of languages, tools and development environments geared to those tasks, and to develop them in a coordinated open-source framework that allows developers both to learn from one another and share partial results, and also encourages development of inter-operable packages so that, for instance, one developer's language module can be incorporated into another developers toolkit for specifying multi-character synthetic plays.

Wherever possible teachers should be encouraged to learn in the same way so that they can start exploring ways of modifying or extending the tools to suit their own interests and teaching preferences, or tailoring them to benefit from the shared local knowledge, cultural variations, or the specific capabilities and interests of their pupils.

(This opposes the philosophy of some commercial 'educational' packages produced by well known companies which assume that the teacher should passively use whatever the package provides and not be allowed to tamper with the code in any way.)

If there is a coordinated open-source multi-site collaborative development process, it should be possible for systems developed for older learners to build on those developed for younger learners, thereby helping to ensure that the process is cumulative and deep rather than just a new collection of relatively isolated things to learn.

But this should *not* involve strong control and a requirement to conform to rigid standards, for it will be many years before anyone knows enough to specify a good set of standards, and in any case there may never be standards equally appropriate to all school environments.

Ideally the results of such an endeavour would be usable all round the world, with teachers in different environments making changes to suit local needs.

And then...

After all this, politicians and educators may perhaps understand learning and development better and grasp the importance of education built on the **five** Rs, namely **reading, writing, arithmetic, and programming**. Universities will be transformed by the new breed of school leavers who will put professors in many disciplines to shame. And far more girls will wish to study computing at University, in many degree programmes, from philosophy to microbiology.

References

A. Sloman. *The Computer Revolution in Philosophy*. Harvester Press (and Humanities Press), Hassocks, Sussex, 1978. (**http://www.cs.bham.ac.uk/research/cogaff/crp/**)