PROGRAMMING WITH A PURPOSE: HANK, GARDENING AND SCHEMA THEORY

Paul Mulholland and Stuart Watt

Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA, UK
Email: P.Mulholland@open.ac.uk; S.N.K.Watt@open.ac.uk
Web: http://kmi.open.ac.uk/people/paulm/hankpage.html

Abstract

Hank is a visual cognitive modelling language designed specifically for psychology students. The aim in designing Hank was to create an experience of cognitive modelling that focused on gaining a new perspective on psychological models rather than programming for its own sake. Recent informal analyses have investigated the effectiveness of Hank in its intended context of use, both as a paper and pencil exercise for individuals, and as a computer based project to be carried out in groups. The findings largely supported the Hank design decisions, and illuminated many of the challenges inherent in designing a programming language for an educational purpose.

1. Introduction

Hank is a cognitive modelling language for psychology students. Cognitive modelling involves building computational models of psychological theories in order to learn more about them, and is a major research area allied to psychology and artificial intelligence. The main problem is that few psychology students have previous programming experience, making conventional languages such as Prolog, which we currently use at the Open University, difficult to teach and learn. Our proposed solution is Hank, a visual cognitive modelling language, designed specifically for the psychology student.

The design of Hank drew heavily on lessons learned in related research areas such as end-user programming, software visualization and the psychology of programming to develop a language that is sufficiently powerful to meet the needs of the students, and also usable (Mulholland and Watt, 1998). Our initial analysis of Hank has been informal, and has concentrated on the specific uses for which Hank is intended. This initial analysis proved very revealing, supporting many of the design decisions, and illustrating the range of complex issues, other than the notation of the language, that impact on the effectiveness of programming as an educational activity.

The next two sections briefly describe the original design objectives, and the Hank environment. This is followed by a description of how Hank has been used, and lessons learned. Finally, improvements made to Hank in light of the findings will be described, followed by an outline of planned future work.

2. Original design objectives

The process of designing Hank began with five key objectives: the language should be specifically aimed at psychology students; it should be usable by non-programmers; it should be usable in groups; it should clearly visualize process; and it should be usable on paper. These will be considered in turn.

- *A cognitive modelling language for psychology students*. Hank is intended primarily to be a cognitive modelling language for psychology students. For this reason it has to bear a clear relation to the cognitive theories and architectures found within the cognitive psychology course studied by the students.

- *Usable by non-programmers*. The majority of the students have no previous programming experience. The language therefore has to be easy to learn and use.
- *Usable in groups*. The students are required to build a cognitive model as part of a residential school project. At the Open University, students attend a residential school for one week during the summer and work on projects in groups of between three and five.
- *Show the process*. Another important requirement is that the language should be able to show its procedural aspects. The currently used language, Prolog, has a complex execution model that can mask procedural characteristics that could be of interest to the cognitive modeller.
- *Usable on paper*. The students' first experience of cognitive modelling is as part of an assignment carried out individually at home. As our students cannot be required to have a computer, the assignment requires the students to write and run small programs on paper.

These requirements combined with a detailed review of the literature led us to the design of Hank, which is described in the next section. A description of the design rationale can be found in Mulholland and Watt (1998).

3. Hank overview

Hank has four main programming constructs: fact cards, instruction cards, questions and the storyboard. Fact cards are what we use in Hank to represent factual information. Three fact cards ("A kind of", "Picnic specifics" and "Picnic defaults") are shown to the top right of figure 1. Fact cards adopt the familiar table structure, used in spreadsheets. The first row is the name of the card (shown in dark grey). The second row gives a label for each individual column (shown in light grey). The remaining rows (called data rows) each represent a related set of symbols. The fact cards in figure 1 form part of a simple model of schema theory. The "A kind of" card indicates which events are members of the picnic category. The "Picnic default" fact card represents typical characteristics of picnics in general. The "Picnic specifics" fact card represents unusual information about particular picnics.

An instruction card represents a procedure that can be used to work something out. An instruction card called "Picnic schema" is shown to the upper centre of figure 1. The top part of the Instruction card, above the double horizontal line, is called the matching box. The matching box defines the goal of the Instruction card. The area below the double line is the process box that describes how the goal can be

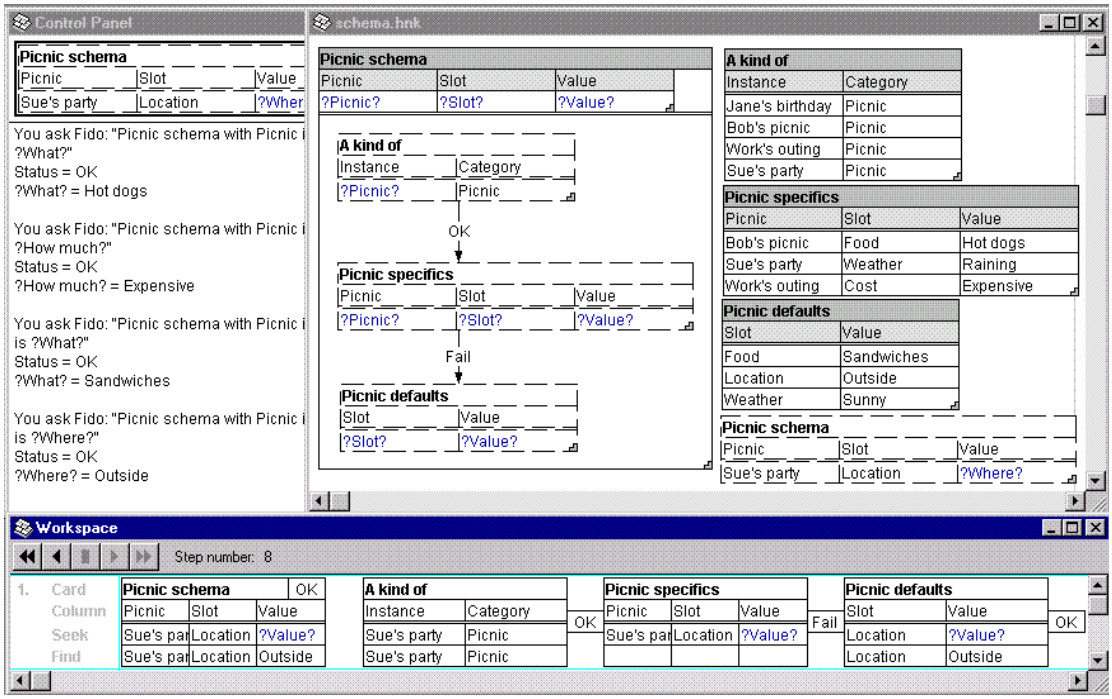

Figure 1. The Hank environment

achieved. The process is defined as a set of questions (shown with dotted lines) connected by arrows. Each question contains Wildcards (referred to as variables in most languages). Wildcards begin and end with a question mark, and in the computer implementation are shown in a different colour. The "Picnic schema" instruction card contains three wildcards: ?Picnic?, ?Slot? and ?Value?. In the "Picnic schema" instruction card, the process involves finding out information about a picnic using the information stored in the fact cards.

Storyboards are a form of comic strip notation used to indicate the serial order of processes and show their causal relationship. This is the visualization mechanism within Hank. The storyboard representation of the execution is shown in the Workspace window to the bottom of figure 1. The storyboard is showing how the Instruction card can be used to infer a location for Sue's party.

A further important feature of Hank is the executive, which is responsible for running the programs. The executive is described as having two parts: Fido and the house rules. Fido is the interpreter that carries out the execution. The house rules form the description of how programs should be executed (i.e. the execution model). Fido follows the house rules when running a program. When students run programs at home on paper, they will take on the role of Fido for themselves. When working in groups at residential school, the job of Fido can be performed by the computer on their behalf.

In the computer version, questions can be sent to the executive using the control panel. It is shown to the top left of figure 1. The control panel keeps a record of previous questions and answers. It is used for asking questions to Fido, and requesting storyboards of the execution. The next two sections describe our experience so far in using Hank for paper-based and computer-based cognitive modelling.

4. Using Hank in a paper and pencil assignment

An initial study was undertaken using a draft version of the paper and pencil assignment. The assignment provides an introduction to cognitive modelling and covers issues related to knowledge representation, category membership and schema theory. The assignment is split into two parts. The first part requires the student to build and run some simple cognitive models. The second part involves writing an essay covering the wider issues surrounding cognitive modelling and artificial intelligence in general. For the study only the first part of the assignment had to be carried out. The programming part of the assignment was sent out to three participants, a student who had previously studied cognitive psychology (including modelling using Prolog), a student who was about to embark on the cognitive psychology course, and an experienced course tutor. The three participants were required to complete the modelling exercises plus a short questionnaire and then return their completed assignment.

Each participant was able to build small programs using fact cards and instruction cards. It was particularly pleasing that the students were able to use the storyboards to run their program, though there was some evidence of students getting lost in the storyboard, and not gleaning what they might have done, with the larger execution histories. Despite this, the study did provide some evidence that two of our design objectives had been addressed, since Hank could be used with success on paper alone, without support from a computer implementation, and by non-programmers.

This exercise was intended to be little more than a sanity check on our part to make sure that the psychological concepts, as portrayed in Hank, were accessible and usable by the students. In this respect, perhaps most significantly, the student with no previous experience of cognitive modelling commented that the experience had given her a much more positive attitude to this area of the course, which she would meet later in her studies. This encouraged us to carry out a more detailed study of using Hank at a residential school.

5. Using Hank at residential school

At a week long residential school, students get the opportunity to carry out a cognitive modelling project for two and a half days in groups of between three and five. The project involves building a rather more complex model than those introduced in the assignment, this time with the help of a computer. For two of the residential school weeks this summer, students were given the opportunity to build their cognitive models using Hank rather than Prolog. Four groups took up the challenge. The authors took responsibility for tutoring the Hank groups.

For each of the participating groups, the cognitive modelling project began at 9am on a Wednesday morning. The process of introducing the language (which involved unlearning certain Prolog concepts) and teaching them how to use the computer implementation of Hank took until lunchtime. The students began their projects in the afternoon session. Groups using Prolog, because they already have some familiarity with the language, usually start their project around 11am. The Hank groups had caught up with their Prolog counterparts by mid morning on Thursday, and had a working model by the end of the project. The groups were able to demonstrate their understanding of the model by explaining to the tutor how it worked, and what it meant psychologically. At the end of the project, each group was interviewed separately. The semi-structured interview comprised eight questions. These are shown below:

- Did the Hank language help you to understand any elements of cognitive psychology? If so which?
- What things did you find easy or straightforward to do using Hank?
- What things did you find hard to do using Hank?
- Can you suggest any improvements to the Hank language?
- What are the differences between Hank and Prolog?
- Have you any ideas on how Hank should be explained to someone who has never seen it before?
- For AI this year, would you have rather used Hank or Prolog?
- In your opinion, should students next year be taught using Hank or Prolog?

Their responses were transcribed by the experimenter. Below, eight key points from the feedback are identified. Their comparative comments of Prolog are not only based on their experience in using Prolog in an assignment earlier in the year, but also on their discussions with Prolog students. One group even invited a number of Prolog students into their room to give them a demonstration of Hank.

1) The removal of the textual syntax proved very successful

A typical comment was:

> "It was much better not having any commas, full stops and brackets"

Although remembering to place these Prolog syntactic constructs in the right place is trivial to someone experienced with the language, it can cause problems when the students are not particularly familiar with the language, and have a number of other issues to consider. One student made an analogy between Prolog and using DOS rather than windows to move files around. Removing the textual syntax had a huge impact on the students, regardless of their experience with computers. On one extreme, there was a student who had never used a computer before, who was soon able to get the hang of constructing cards and running the program. On the other hand, there were some students who used spreadsheets extensively as part of their job, and felt immediately comfortable with the notation and how it worked.

2) The students were able to relate their Hank programming to the course materials

Students were able to draw appropriate links between their Hank project and the course materials. This was found in comments such as:

> "Hank emphasises the model not the computer programming"

In particular, the link to schema theory was very clear. This was illustrated in comments such as:

> "It has a direct relation to schema theory, far more so than Prolog, due to the column names"
>
> "It distinguished between specific and general information in schemas"

The tabular structure of facts in Hank is very close to the way schemas tend to be represented in cognitive psychology textbooks, as a set of slots and values. This is illustrated in figure 2.

Actor:	John
Act:	ATRANS
Object:	necklace
Direction TO:	Mary
Direction FROM:	John

Schema	
Slot	Value
Actor	John
Act	ATRANS
Object	necklace
Direction TO	Mary
Direction FROM	John

Figure 2. On the left, a typical schema representation from Eysenck and Keane (1995) and (right) how the schema could be represented in Hank.

3) Students were able to understand each other's programs

As the students could identify how their programs related to psychological theory, they were more able to see connections not only between their program and the course texts, but also between their program and other Hank programs. This was illustrated in comments such as:

> "I could see the analogy between the blocks world program [their project] and the families program [a warm-up program tried out at the beginning of the project]"

> "You could see that they were similar on a deeper level"

> "I can see how we could do other models using Hank, such as language understanding"

The students were able to understand other programs because the programs were no longer being compared in terms of syntactic similarities, but rather in terms of the theories that were being modelled. The students were even able to see how Hank could be used in domains other than cognitive modelling. This is nicely illustrated by a comment from one student who worked as a gardener.

> "Hank could be used in real life. As a gardener I could use it to keep a database on trees: level of shade and light, growth rate. I could write a rule to find the best conditions, a bit like our cousins program"

This comment is particularly encouraging as the student thinks not only about how Hank can be applied in another domain, but also how the Hank program would have similarities to another program (called the 'cousins' program) that the group had developed when familiarising themselves with Hank on Wednesday morning.

4) Students could understand the process as well as the result, though syntonicity did not work as expected

Prolog has a very complex execution model, therefore even if the students' program is working correctly, they may not know why. In Hank, the students could clearly see the connection between their program and how it worked. This is illustrated in comments such as:

> "I liked the "Ask fully" option [used for generating storyboards]. It was easy to see where we had gone wrong"

> "The process of getting to the answer can be clearly seen"

An aim in our design was to use syntonicity to help students identify with the process. It had been our intention that students should identify with Fido, the interpreter, though Hank became the identified character.

> "It's good being able to work through the program but I'm not sure about Fido though"

> "We liked to personalise Hank, by saying Hank does this, Hank does that"

We will return to these issues later in the paper.

5) Their approach was confident and exploratory rather than nervous and tentative

When Prolog students reach an impasse it is very difficult for them to explore the language in an attempt solve the problem without the tutor's help. Hank students were more keen to explore and try different things out when their model was not working correctly.

"You are more willing and able to explore in Hank"

"Basically, Hank is fun. You can fiddle with it and see what it does"

6) The learning curve was more linear and determined by theoretical rather than programming concerns

Because of the more exploratory nature of the language, the learning curve was more linear. This contrasts with Prolog which tends to comprise one or two large jumps.

"You can test and change you program. Prolog either works or messes up"

"Hank appears intuitively easy and then gets hard later on. With Prolog, you start thinking it's impossible, and then it gets harder"

"Hank goes from easy to hard, and then to easy. Prolog looks hard"

7) Some minor interface issues need to be resolved

Some interface issues were raised by students, concerned with the way objects are selected and moved.

"The distinction between normal questions, and questions inside the rule is unclear"

"Sizing boxes and columns was sometimes difficult"

"The scrolling was quite bad. It was too slow"

These issues have already been rectified in the current version of Hank.

8) Working with Hank was enjoyable

Learning with Hank was enjoyable as well as effective.

"Hank is pleasing to look at"

"Hank is fun"

From a tutor's perspective, a number of issues were raised concerned in particular with the relationship between the Hank language and the educational experience as a whole.

1) Hank offers many alternative solutions to the same problem, and many routes to them

A major difference between Hank and Prolog is that in Hank, the same problem can be solved with equal effort in a number of different ways. This requires rather a lot of thinking ahead on the part of the tutor, who has to discern which solution the students seem to heading toward, and how they should be guided there. This illustrated the need to rethink the relative pedagogical benefits of alternative paths. We are currently using the educational walkthrough technique (Lewis, Brand, Cherry and Rader, 1998) to chart how the programming process undertaken by the students links to educational objectives.

2) Time devoted to meta-talk about the design itself and how it compared to Prolog

The students were not treated as naive subjects in an experiment. We often got into discussions with the students on why Hank was designed in such a way, and how that related to concepts they had studied in their psychology course. For example, a discussion took place comparing programming in Prolog and Hank, in the way that homomorphic problems are compared in their course text on human problem solving (Kahney, 1993). Another student with a knowledge of HCI suggested how Hank could be understood using the notion of affordances as described by Norman (1993). This conscious

decision to encourage the students to be joint participants in a design process (Bannon, 1991), rather than passive subjects, led to many interesting suggestions and observations.

3) Motivation, enjoyment and teamwork

The students were highly motivated, exploratory, and keen to try things for themselves. Hank became a tool for thinking with (which is what cognitive modelling languages should be) rather than a pedantic machine holding them back (Christiansen, 1997). As commented by the students, it was far easier to understand Hank programs written by other people, compared to Prolog. This led to some rather complex group-working arrangements, where different members of a group would work on different parts of the program, or different solutions to the same problem, before coming back together to compare their ideas. We were amazed by how smoothly this approach worked.

Overall, using Hank at residential school in place of Prolog was found to provide a very successful educational experience. Not only were the students able to complete the project (a feat in its own right given they were seeing the language for the first time), they had also clearly learned from the experience. They were also able to draw strong links between the Hank project and key issues and theories from within the course.

6. Changes to Hank

The main changes taken in light of the work so far were downplaying the role of Fido, simplifying the structure of the house rules, and some terminological changes.

Shift in focus from Fido to Hank

Our original notion of how to use syntonicity (Papert, 1993; Watt, 1998) to encourage identification with (leading to an understanding of) the program did not work as expected. The students were better able to understand the process, but tended to identify with Hank as a whole, rather than Fido sitting inside.

The role of Fido in Hank has certain parallels to Searle's (1980) Chinese Room argument. Searle told a story of a non-Chinese speaking man sitting inside a box passing (what where to him) meaningless symbols out of a box in response to other symbols that entered, according to a set of rules. A Chinese speaking person outside the box believes they are communicating with someone who understands them. The story is told to question the artificial intelligence approach by showing there is no understanding in the box. Fido performs a similar job, showing there is "nothing special" going on inside Hank, but as in the Chinese Room, this does not lead to identification with the character inside who has no understanding of what is happening.

Now that computers are becoming ever more commonplace, showing people that there is nothing mystical about how the computer works is becoming less important, and therefore there is less need to illustrate this point. However, there was evidence of syntonicity, Hank being the target. This is because syntonicity tended to be used as a shorthand form of communication for what was happening. Students often made comments of the kind "Hank is trying to do this". The students appreciated the difference between the language (Hank) and the interpreter (Fido) but did not want to explicitly draw on this when explaining what their program was doing. For this reason, we have since decided to play down the role of the interpreter in the execution model we present to the students.

Simplification of the house rules

So far, the house rules have been presented as a fine-grained recipe of how programs should be run. Although the students need to be given a detailed story of how the programs are run, the current house rules seemed to make it difficult for the students to see the wood for the trees. In our revised version the presentation of the house rules has been simplified to just a small number of bullet points, each with a supporting paragraph of what work that bullet point involves. In the new approach we are designing for growth in competence. As the students become more familiar with Hank, the details pertaining to each bullet point become operationalized (Kutti, 1997). In the original house rules, there was no scope for students to operationalize parts of the process as their familiarity increased.

Changes to terminology

A few minor changes in terminology have also occurred. Wildcards have been renamed as variables. This is to remove any possible confusion with Fact Cards and Instruction Cards. In order to help students in designing an Instruction Card, the top part of the instruction card is now named the "wish box" rather than the "matching box". This terminological change developed out of discussions with students at residential school, where they designed their instruction cards, by first typing into the top part of the instruction card what it was they wanted to find out, and then moving on to work out how it could be produced. Finally, Fido has been renamed the question processor, to encourage syntonicity with Hank as a whole.

7. Further work

So far, Hank has been used to allow students to gain a useful first experience of cognitive modelling. We now wish to develop Hank further to investigate how psychologists can be supported in developing more complex models. Some current and planned developments to Hank involve extending support for cognitive architectural features (Anderson, 1984; Newell, 1990) concerned with issues such as capacity and latency in memory. Other new features are to support book-keeping and the initialisation of models, such as mathematical functions. These developments, though, are not just restricted to adding more primitives to the existing Hank design. We also have plans to build a model level layer where components of the Hank program can be tied to particular modules within the box-and-arrow style models prevalent in cognitive psychology.

The further development of Hank is linked to an ongoing programme of evaluation, and of developing real and substantial cognitive models both informally and through more formal walkthroughs (Lewis and Rieman, 1994), to ensure that the language can handle models of a rather larger scale than those met in the studies we have described. This development programme is constrained within the existing Hank language framework, so that its benefits are maintained. This is giving us a good understanding of which changes to make, and how to make them safely.

8. Conclusions

Our initial analysis has largely supported our initial design of Hank in enabling students to develop cognitive models on paper and on the computer, and to use those models to illuminate their understanding of cognitive psychology. Let's look at this in a bit more detail, returning to our original design objectives, set out in section 2:

- *A cognitive modelling language for psychology students*. Both studies showed that Hank helped to cut through to the psychological concepts of the model being built, bypassing messy programming.
- *Usable by non-programmers*. Again, both studies provided evidence that non-programmers could use Hank, and at the residential school, the computer implementation was even used successfully by students who had never used a computer before.
- *Usable in groups*. Groups worked well in the residential school study. An unexpected change was that because Hank programs were generally easier for people to understand, the collaborative processes were very different to those found with Prolog groups. Hank groups tended to use more complex group structures, compared to Prolog groups' typical 'all for one, one for all' format.
- *Show the process*. Both studies provided some evidence that the process was more apparent. Although students found large storyboards on paper difficult, with the computer implementation, there was clear evidence that they did begin to understand how their models had worked more fully than they had with Prolog, which tended to work by 'magic'.
- *Usable on paper*. The home-based, paper and pencil study, confirmed that Hank could be used on paper, at least to a limited degree, and without a tutor being present in person.

Perhaps most surprising, though, has been the sheer engagement that Hank managed to bring to the students at the residential school. On the whole, they simply revelled in it, and many found it "fun" compared to Prolog. Perhaps the motivational benefits of this engagement are also playing an important role in the students' successful work with Hank. As a project, Hank is still proving a rich source of useful research for the future.

Acknowledgements
We are grateful to the students and tutor who took part in the analyses presented in this paper.

References

Anderson, J. R. (1984). *The architecture of cognition*. Harvard: Harvard University Press.

Bannon, L. J. (1991). From human factors to human actors: The role of psychology and human-computer interaction studies in system design. In J. Greenbaum and M. Kyng (Eds.). *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Lawrence Erlbaum.

Christiansen, E. (1997). Tamed by a rose: computers as tools in human activity. In B. A. Nardi (Ed.). *Context and Consciousness: Activity Theory and Human-Computer Interaction*. Cambridge, MA: MIT Press.

Eysenck, M. W. and Keane, M. (1995). *Cognitive Psychology: A Students' Handbook*. Lawrence Erlbaum Associates.

Kahney, H. (1993). *Problem Solving: Current Issues. Second Edition*. Buckingham, UK: Open University Press.

Kutti, K. (1996). Activity theory as a potential framework for human-computer interaction. In B. A. Nardi (Ed.). *Context and Consciousness: Activity Theory and Human-Computer Interaction*. Cambridge, MA: MIT Press.

Lewis, C., Brand, C., Cherry, G., and Rader, C. (1998). Adapting user interface design methods to the design of educational activities. In Proceedings of *Human Factors in Computing Systems, CHI '98*, Los Angeles.

Lewis, C. and Rieman, J. (1994). Task-centered user interface design: a practical introduction. Shareware book, available at ftp://ftp.cs.colorado.edu/pub/distribs/clewis/HCI-Design-Book/.

Mulholland, P. and Watt S. N. K. (1998). Hank: A friendly cognitive modelling language for psychology students. In Proceedings of *IEEE Symposium on Visual Languages, VL '98*, Nova Scotia, Canada.

Newell, A. (1990). *Unified theories of cognition*. Cambridge MA: Harvard University Press.

Norman, D. A. (1993). *Things that make us smart: Defining human attributes in the age of the machine*. Reading, MA: Addison-Wesley.

Papert, S. (1993). *Mindstorms: Children, Computers and Powerful Ideas, Second Edition*. Hemel Hempstead, UK: Harvester Wheatsheaf.

Searle, J. R. (1980). Minds, brains and programs. *Behavioural and Brain Sciences*, **3**, 417-424.

Watt, S. N. K. (1998). Syntonicity and the psychology of programming. Collected Papers of the *Psychology of Programming Interest Group*, The Open University, Milton Keynes, UK.