Mental models to support competence in computer programming

Dr. Richard Millwood

Trinity College Dublin

richard.millwood@tcd.ie

Mental models

These are the mind's 'mechanisms' for explaining and predicting phenomena. The idea originated with Craik in World War II and elaborated by Johnson-Laird (2004) amongst others (Simmons, Wikipedia).

This is not the same thing as modelling phenomena in mathematics or in computer programs - such simulation models are expressions in external languages, unlike mental models, which are private to our minds: interconnected, fluid, faulty and ultimately unknowable. When we create and communicate external expressions in natural or formal language, this leads to the possibility of proof, execution and formal reasoning in a shared world of knowledge. But these external expressions aren't mental models: they are in a linguistic form that can be interpreted by others and in formal cases, by machines.

Nevertheless he unknowable internal mental model remains a useful notion, and is applied here to the design thinking needed for effective courses, materials, pedagogy, software, assessment to teach programming.

This mental model allows the learner to reason about the problem itself - it may develop as the learner combines problem solving and design to make a solution. Sometimes prior knowledge can help; for example, Papert would argue that children enjoy, are competent and have mental models about the way their body can move in the physical world. If a problem is aligned to such competence, they can more effectively debug their program (body syntonic) and feel engaged with the challenge (ego syntonic).

This mental model is about the parts of the language - the distinctions between different linguistic components and their connection to create programs. Scratch supports this mental model by categorising statements and thus offers recognition rather than recall. It also reinforces appropriate syntactical combinations, so that the focus is on their meaning, in isolation and in combination.

problem comprehension

How do I represent key problem variables & processes?

What data structures, algorithms & statements are available to map on to the problem?

programming language

Does the problem fit the microworld?

How does behaviour in the microworld offer a solution?

interative development environment

When statements are executed, what is modified in what order?

How are changes in the notional machine seen?

microworld / domain

What outputs and changes will happen in response to the program?

What parts of the notional machine link to the microworld?

notional machine

The microworld is the concept of a limited 'space', designed to suit a particular class of problems and usually with an 'object to think with'. The turtle geometry microworld is the most famous, but not the first in Logo. Before that came sentence construction using lists of words to manufacture amusing nonsense! Making a mental model of a microworld's affordances allows the learner to map solutions to problems and to relate to the notional machine and the programming language, within a limited but meaningful domain.

The notional machine is a mental model concerned with the variables, computer memory (for data and program), 'program counter' - a hidden variable that determines which statement is executed next and thus flow of control. It is much more complex with Scratch than in the past, since multiple parallel process are readily designed using sprites. The design of solutions in this way can be quite different from that made with single process thread programming, but makes the mental model challenging.

The mental model here is of a complex user interface to understand and write programs, manage program files, debug programs and produce results. Sometimes it spans several computer applications, such as an editor, file manager and version control, and sometimes it csn be combined in one place, as with Scratch. It is the interactive development environment (IDE) which can help or hinders the user in forming the mental models of programming language, notional machine and microworld through visualisation and interactivity.

References

Johnson-Laird, P.N. (2004) The history of mental models. In Manktelow, K., and Chung, M.C. (Eds.) Psychology of Reasoning: Theoretical and Historical Perspectives. New York: Psychology Press. Pp. 179-212.

Simmons, M. (2017) The Top 12 Most Useful & Universal Mental Models, medium.com/the-mission/this-is-exactly-how-you-should-train-yourself-to-be-smarter-infographic-86d0d42ad41c

Wikipedia (2018) Mental Model, en.wikipedia.org/wiki/Mental_model

Waite, J. (2018) Personal Communication