FITTER00

Fitter, M., Green, T.R.G.; When do diagrams make good computer languages?; International Journal of Man-Machine Studies 11.2 pp. 235-262.

A good paper discussing the major issues in the use of diagrammatic notation in programming. Heavily illustrated with examples and references to experimental results. They push the point that we do not know how to do it right, and that experimentation is necessary.

The authors separate coding notations into perceptual and symbolic. Symbolic notations are the traditional text string notations, and other notations, where the meaning has to be inferred from a symbolic encoding. In a diagrammatic notation, the meaning should be perceptually obvious (i.e., should not require high level mental processes to decode). Hence, diagrammatic notations are considered perceptual notations.

The main body of the text deals with five key principles for designing and selecting perceptual notations. These are: only relevant information should be perceptually coded; the user should be restricted to comprehensible forms; important information should be redundantly recoded; the coding should reveal the underlying processes and the diagrams must be revisable. Each of these principles is discussed in detail, and illustrated with many examples.

The paper provides a very good introduction to diagrammatic programming notations.

"Our conclusion is that if a graphic notation can reveal the structure inherent in the underlying data, or the process by which entities are manipulated, then it will be superior to a linear symbolic language." p. 255

:Different programs should be perceptually as different as possible." p. 259