SENGLER83

Sengler, H.E., "A Model of the Understanding of a Program and Its Impact on the Design of the Programming Language GRADE", Psychology of Computer Use, T. R.G. Green et al. (eds.), 1983, pp. 91-106 - Program Structure and Representation. This paper presents a cognitive model of program understanding by experienced programmers, and outlines the design of a graphical programming language based on principles derived from the model. According to this model of program understanding, a programmer reading a program maps the text into a mental structure called the "internal program". This internal program is a net whose nodes, or components, are the semantic units of the programming language used in the program, and whose arcs, or relations, are semantic relations between these units. The process of understanding involves separating the program into a hierarchy of intellectually manageable portions. For any given portion, the programmer findsits components, relations, inner portions, and outer relations (i.e. the portions interface to the outside). He then proceeds to understand the inner portions and outer relations between these units. The process of understanding involves separating the program into a hierarchy of intellectually manageable portions. For any given portion, the programmer finds its components, relations, inner portions, and outer relations (i.e. the portion’s interface to the outside). He then proceeds to understand the inner portions and outer relations by reducing or enlarging the portion under consideration. After these have been understood, he associates the semantics of components and relations, recalls the semantics of inner portions and outer relations, and evaluates the resulting semantics of the portion by combining the consituents’ semantics and imagining the effect of the portion on the program’s state space (i.e. the set of entities addressed). In addition, the resulting semantics is abstracted, giving a concept of the function of the portion. Three properties of a program are relevant for understandability: The syntax and semantics of components and relations, basically defined as language elements in the programming language, affect the ease of finding, associating and evaluating. The syntax of language elements should be distinct, compact, conspicuous and easy to follow, and their semantics should be simple in terms of information content and relations to other language elements. It is argued that the use of a graphical representation fulfils these requirements. The representation and composition of portions affects finding and evaluating. Although this is mainly determined by program design methods, the programming language should provide the necessary means to structure the program as a hierarchy of portions with clearly defined boundaries, interfaces, and constituent components and relations. Graphical representation helps achieve this objective. The representation, structure and changeability of the program’s state space affects evaluating. To simplify this task, the state space should be represented in the program and statically structured according to the program structure, and changes to the state space should be sequential and deterministic. A programming language GRADE has been defined and a pilot implementation produced to illustrate the above properties derived from the model. A GRADE program is a data processing machine consisting of parts that produce, transform or store data. These parts are connected by lines that allow the transport of data and control pulses. The semantic concepts of the language are represented graphically: A circle represents a store, which accpets or produces data. A rectangle represents a processor, which fetches, transforms or sends data. A solid directed line represents a data line and a dashed line represents an activation, or control, line. A trapezoid represents a branch within an activation line. Preliminary observations indicate that GRADE is more suitably used on a high level of the program structure, and that the programmer constructing a GRADE program is urged to develop a simple program structure through the visibility of the structure in the graphic representation. Although more research is required to determine the utility of a graphical programming language such as GRADE, the model of program understanding presented in this paper is generally applicable to other classes of programming languages.