TOOLS FOR MODULAR PROGRAMMING: FINDING OUT WHAT'S NEEDED

John B. Goodenough
SofTech, Inc.

Modularity, in our view,[1] deals with how to structure programs so that specified purposes are more easily attained. Our current work deals with three purposes -- modifiability, reliability, and efficiency -- and seeks to identify what structural properties, tools, and language features facilitate the attainment of these purposes. Our approach is to study the structure of three actual programs to see how their structure makes the programs more or less modifiable, reliable, and efficient. The first program is a 1,000 statement PL/I program written in top-down style with levels of abstraction. The second is a large circuit analysis program (AEDCAP). The third is a general purpose package for processing list structured data in a fashion that permits changing details of the data structure representation without revising high level algorithms operating on the data. The case study results will be used to assess the potential utility of possible programming tools and language features.[2]

1. System Organization Technology: An Analysis of Modularity. Available from SofTech.
2. This work is supported by Army contract DAAA 25-72C-0667.

A METHODOLOGY FOR PRODUCING RELIABLE SOFTWARE SYSTEMS

Carlos J. Lucena*
University of California, Los Angeles

A research program is underway with the purpose of developing a system to integrate the functions of Design, Programming, Debugging and Testing (DPDT) for achieving software reliability. Although we are aiming at developing an actual formalized system for this purpose, emphasis is being placed on the isolation of general principles that can be applied to a variety of programming environments. We are defining a model of computation through which the design phase will be formally described.[1] The model will be required to handle a basic definition of modularity besides having features to be integrated with: a class of Algol-like languages (in which programs are to be written), the debugging facilities defined in the language, and a class of program testing strategies. Extensions to a programming language will be defined so as to make it support the requirements of the design model[2] and to provide debugging facilities compatible with the design phase and measurements. The measurements will be handled by the systems testing phase for the establishment of some quantitative indices of test completeness. The testing activity will in turn reflect on the design and provoke iterations until a "sufficient verification" of the produced software is met.

[1] Dennis, J.B., "Modularity," Advanced Course in Software Engineering, Springer-Verlag, 1973

[2] Parnas, D.L., "A Technique for Software Module Specification with Examples," Comm. ACM, 15, 330

[3] Lucena, C., "Modeling, Measurement and Software Reliability," Internal Memorandum #115, Computer Science Department, UCLA, April 1973

* On leave from PUC - Rio de Janeiro

ENVIRONMENT PRIMITIVE OPERATORS IN PROGRAMMING LANGUAGES

Richard O. Clark
James F. Leathrum
University of Delaware

An extensible language implies that the fundamental semantics of a programming language are included in the base language. Those basic operations which are not so defined cannot be produced by extension. One aspect which has not been decomposed into a set of primitive actions is that of environmental operators. These are operators which describe the binding between a symbol, its value-structure, and a value. As a result, the execution environment of the base language applies to each extension and the style of the language remains fixed. The same remarks also apply to formal semantic descriptions of programming languages. In this paper a value storage mechanism, a value reference mechanism, and a set of binding operators are proposed which permit the execution of a program in the context of a FORTRAN, ALGOL, PL/1, or APL environment.

INVESTIGATION OF THE MECHANISMS FOR INTERPROCESS COMMUNICATION

Jan Polek and Imtiaz Ahmad
University of Ottawa

From a study of the computer systems operating in a multiprogramming environment, a model for the interrupt activity is derived. The basic components of the interrupt activity are identified and a formulation is developed to express process switching in terms of these components. The systems studied include, in particular, the IBM 360/370, Burroughs 5000/6000/6, PDP 10/11/8, Nova 1200, etc. Some important aspects of interrupt structure, such as, response time, overhead and saturation are examined and compared. The results indicate that while logical functions[1], such as, interrupt acknowledge, saving, servicing and post processing are easily identifiable, their hardware implementation exhibit variations which have significant influence on process switching flexibility[2]. The use of interrupt mechanisms in scheduling of processes in multiprogramming systems is illustrated with examples. Problems of suitable functional modules for processor multiplexing are discussed. The results of the aforementioned study are used to suggest a suitable representation of the mechanisms for interprocess communication.

[1]Borger, E.R., "Characteristics of priority interrupts," Datamation, 6, 31-34, June 1965.

[2]Dijkstra, E., "Hierarchical ordering of sequential processes," Acta Informatica, 1, 115-138, 1971.

USE OF SNOBOL4 AS AN INTERACTIVE LANGUAGE FOR ARTIFICIAL INTELLIGENCE RESEARCH

Dewayne Hendricks
University of Michigan

SNOBOL4, a very good general purpose string manipulation language, has been made little use of in the past by researchers in the area artificial intelligence. Some of the possible reasons for this could be due to the akward syntax of the language (from the point-of-view of a procedure-orinted programmer) and the large memory requirements of the language in most of its implementations. Modifications have been made to SNOBOL4 as implemented in a virtual memory environment on a 360/67 under MTS, to tailor the language to an interactive environment. Also, a supervisor program has been implemented which asts as an interface between SNOBOL4 and the user and provides him with many of the features which are available in PLANNER and CONNIVER. These modifications have resulted in a language which is quite useful for AI research under MTS.

ON-LINE DATA ANALYSIS AND GRAPHICS USING BASIS-70

C. Claydon, R. Krohn, A. Fish
Battelle Memorial Institute

The BASIS-70 user interface language for interactive information storage and retrieval has been described earlier.[1] This language has been extended to allow the user to perform arithmetic and functional operations on items of the retrieved records. Also, the extended language allows the user to format as desired a plot on an interactive graphics terminal and to request 35mm film copies be produced on the Stromberg Datagraphix 4060. Several examples will be presented of the following steps: (a) interactive selection and retrieval of a subset of records from a large data base, (b) polynomial regression of data base items, and (c) plot of regressed polynomial.

[1]John B. Fried, "The BASIS-70 User Interface", Interactive Bibliographic Search: The User/Computer Interface, Edited by D. E. Walker, AFIPS Press (1971).
