STANDARDIZATION OF SOFTWARE FOR CACSD

Panel Discussion

Chairman: Mohamed Mansour
Swiss Federal Institute of Technology (ETH),
Zurich, Switzerland

Members: Derek P. Atherton
University of Sussex, Brighton, U.K.

Michel A. Floyd
Integrated Systems Inc., Palo Alto, California USA

Hartmut Hensel
Technische Hochschule Darmstadt, FRG

Magnus Rimvall
Swiss Federal Institute of Technology (ETH),
Zurich, Switzerland

Keywords: CAD, Software, Standardization

INTRODUCTION

The aim of this discussion is to point out roughly what kind of standardizations can be made to CACSD software and how can this be realized. Hopefully at the end of the discussion we can find a way inside IFAC to begin work in this direction.

During the last decade, a large number of CACSD packages have been developed at different universities and in the industry. These programs/packages are often very specific, unflexible, unhandy in their man-machine interfaces, or are constructed to meet special requirements of the hardware or the operating system software (Hensel). In the following, standardization in the following areas are addressed.

1. Programming languages

The introduction of new powerful programming languages like PASCAL, MODULA-2 and ADA may facilitate the wide use of some program packages due to the transparent program-structures, the high level portability and the computer independent user-surfaces. However the acceptance of these new languages is rather poor due to the excessive effort spent on programs written in the old languages (Hensel). To counteract the increase in language confusion (Fortran, Ada, Modula, Pascal, Prolog ... etc.) a command language standard and a data format standard for the exchange of data between different control packages is necessary (Rimvall).

2. Simulation languages

Atherton points out that simulation is the core of a CACSD package. The system description software should be closely linked to a simulation language. Some simulation languages allow linearized models to be produced, although in general one can see the need for other forms of reduction of the simulation model. Floyd also considers simulation as the core of a CACSD package and therefore there is a need for a standardization of dynamic data representation.
Hensel sees that there is a large number of simulation languages for continuous as well as discrete systems. Standardization in this area should reduce their number and support the development of a few more general simulation languages.

Rimvall considers the CSSL'67 standard to be a step in this direction. The development of simulation languages after this standard is typical and should encourage standardization in the area of CACSD. However, to avoid duplication of work, any standardization group in CACSD software should work in close conjunction with the standardization group within the simulation community (IMACS and SCS).

3. GENERAL PURPOSE OPERATING SYSTEMS

The most promising candidate for standardization of general purpose operating systems is the UNIX-system which offers a computer-family independent operating system surface and programming environment. Major drawbacks of this operating system are the rather slow system reaction performance, the large variety of extensions to the basic standard and the absence of a good real time capability (Hensel). Floyd gave UNIX as an example how standardization has failed in the area of operating systems, which would likely be the case also in CACSD.

4. GRAPHICAL SOFTWARE

One very popular representative of graphical software which already became ISO-norm is the graphical kernel system (GKS). It offers a uniform user data interface which is almost completely independent from the installed hardware (achieved by writing special drivers for each connected device). The GKS-system offers different capability levels so that the system can be tuned with respect to the used device (Hensel). Atherton states that the problem is greatly compounded when moving software to the new generation of powerful workstations with colour graphics, multiple windows etc.

5. MODELING AND DESIGN DATA BASES

In control engineering the engineer's design is eventually transformed into real-time control software or hardware. The key to the long term success of CACSD lies in the automation of the real-time programming process, the critical software component will be the design data base. (Floyd) A fully evolved database should be able to describe any dynamic model. The component library should include linear, nonlinear and logical elements. Discrete-time attributes such as sampling rates (including multiple rates) and delays must be clearly defined. A complete data base can provide accurate and reliable communication among control engineer and real time developers. Floyd gives as example "System Build" used by MATRIX-x.

Atherton also states that entering the description of the system is the starting point for standardization. Any package must be capable of accepting the complete system description, which can presumably include nonlinear and possibly time varying elements, using several possible techniques and be able to convert between them (state space, transfer function, connection statements, block diagrams, signal flow graphs). The files holding the system description data of the appropriate form should then be readable directly into analytical design routines. Obviously it should also be easy to make changes to the system description, either because errors have been made during the entry or for the purpose of studying the behaviour for different parameter sets.

6. USERS INTERFACES

This is the part of CACSD package which is visible to the user. Floyd states that the design of user interface for CACSD software is fairly subjective and can not be subject to standardization. Standardization of the user interface would only stifle creativity in one of the most rapidly evolving areas in computer science. On the other hand Rimvall sees that a standard interactive command-language is needed to enable the exchange of macros and to unify the user interfaces of different packages. Such a standard can and should be described consistently using a set of "BNF"-productions for the syntax with textual additions for the semantic details. Any standard on this level will be hard to enforce on already existing packages, but is likely to be followed by any new packages. Hensel sees that the dialog interface (at its program end) must be standardized in order to achieve simple structuring of dialogs during the program development phase. The dialog itself should be uniform and easy to learn. In the case of command or menu-technique an automatic adaptation of help features dependant on the user's advancement can be very convenient. A future aspect to be considered might be how to change the man-machine to a man-
model-communication, which allows the user to work directly on a model-base without the need of using related system data. This would also reduce the required level of abstraction. Atherton considers the standardization of user interface mainly in view of the standardization of system description as mentioned before under 5.

7. DATA FORMAT STANDARDIZATION

A standard data format (supporting all common control structures) for the exchange of data between different control packages over external files is very much needed. Such a standard is easy to enforce: each program can have several input/output modes, one such mode can support the standard while the other modes are left as they are (Rimvall).

8. INTERFACES TO NUMERICAL ALGORITHMS

Floyd states that standardization of numerical algorithms would be difficult to achieve in a non-commercial environment. However, Rimvall sees the possibility to specify how the algorithm is to be accessed interactively. This facilitates the exchange of algorithms between different CACSD-packages, but also couples the subprogram standardization with the interactive command-language standardization, and is not totally free from implementation restrictions [e.g. the supported data structures of the implementation language and target machine limitations].

CONCLUSIONS

The discussion in the session has shown that there is a necessity to standardize some aspects of the software for CACSD, mainly in the man-machine and program-program interface area. It was felt that IFAC should establish a working group in this area. The organizer of the session has already taken steps in this direction and it is hoped that soon there will be a result of the efforts in this direction.

M. Mansour