THE DEFINITION, DESIGN, IMPLEMENTATION AND USE OF A COMPREHENSIVE SPORTS BIOMECHANICS SOFTWARE PACKAGE FOR THE ACORN ARCHIMEDES 440 MICRO-COMPUTER

BARTLETT, R.M.
Sports Biomechanics Laboratory
Crewe and Alsager College of Higher Education
ALSAGER, Stoke-on-Trent
ENGLAND

INTRODUCTION

The objective of this presentation is to report the development and use of a comprehensive sports biomechanics analysis software package. The software was intended for use with the laboratory's Archimedes 440 microcomputers. These incorporate a 32 bit RISC architecture microprocessor and use pipelined execution of the three cycle instructions (fetch, decode, execute). With a clock rate of 8 MHz and a very short interrupt latency of less than 1 ms, this makes for a very fast machine.

The development of this package was undertaken because of the lack of commercially available sports biomechanics software for this microcomputer. It was intended to be used in the laboratory's extensive research programme and in the teaching of undergraduate courses in sports biomechanics. The package had to deal with the following:

* 8 channel EMG signals obtained from MIE 1000 or 8000 gain skin mounted preamplifiers, with or without telemetry (MIE MT8) and further amplification. Signals are analogue/digitally converted (Microlink 12 bit A/D converter) and transferred to the computer using the IEEE bus at sampling rates per channel of up to 3000 Hz;

* a Kistler 9281A11 piezo-electric force platform with an 8 channel SN9861A charge amplifier unit. A/D conversion as above;

* video coordinate digitising based around the Watford electronics monochrome and the Arvis colour frame grabbers/ video digitisers;

* cinematographic coordinate digitising using the TDS HR48 and A3 digitising tablets.

The software package was developed by adapting the principles of good software engineering as described by Sommerville (1989) to a single person project and the development lifecycle is represented by Figure 1. The most important software attribute was considered to be maintainability as there will undoubtedly be adaptations to changing requirements during the operational life of the package, and these adaptations may be the responsibility of someone other than the author of the existing package.

REQUIREMENTS DEFINITION

Because of the non-criticality of the project and its development by one person, a formal specification was not considered necessary. However, the requirements of the package were clearly defined using natural language and in a format which ensured the traceability of the requirements through the later stages of design and coding. The requirements definition included both the functional requirements of the system, i.e. the services expected of it by its users, and non functional requirements which defined any operational constraints. The functional requirements were based on a comprehensive search of the relevant literature (e.g. Bartlett, 1989; Dainty and Norman, 1988; ISEK, 1980) and consultations with potential users.

Informally stated, the software provides:

* EMG data acquisition and storage and temporal analysis, as recommended by ISEK (1980), incorporating the

raw EMG, full- wave-rectified EMG, mean rectified EMG, integrated EMG and differentiated EMG (with spectral analysis to be added in the next phase of development);

* force platform data acquisition and analysis of forces, load rates, points of force application, moments of force, relevant kinematic parameters and force vectors;
* full kinematic analyses, including joint angles and coordinates and mass centre coordinates and first and second time derivatives of all of these, as well as stick figure representations of both planar and spatial motions based on digitised coordinates obtain from images from one and two cine or video cameras respectively. Data is smoothed and differentiated using cross-validated quintic splines. Three dimensional reconstructions use the DLT algorithm with allowance for linear lens distortion (Karara, 1980).

SOFTWARE DESIGN

Because of the nature of the package, an essentially top down functional design strategy was used with different levels of abstraction. This was undertaken to ensure that the design captured the requirements and to formally document each component of the software so that it could be easily implemented. The design proceeded through the use of data-flow diagrams as specified by Constantine and Yourdon (1979) (e.g. Figure 2). These were supplemented by a data dictionary to represent all data and process definitions. Control flow was modelled by finite state machines (e.g. Fig.3). The partitioning of the data flow diagrams was stopped when the processes represented were easily described by a single page minispec with a clearly defined function or related group of functions.

The user interface was designed to suit the needs and abilities of the users, to be consistent throughout the package and to be visually attractive. The Archimedes 440 window manager provides a versatile interface. However, novice users, such as many of the laboratory's undergraduate students, lack familiarity with windows environments and would need to look up instructions on dragging, selecting, scrolling etc. before being able to use the system.

For this reason, a simple menu system was preferred to the windows environment available on the Archimedes. This was designed to allow for two simple types of user response:

* a choice from a list of options using the mouse to point to and select the required option;
* very short text inputs from the keyboard e.g. to define a file name.

In both cases, it was made impossible for the user to make inappropriate choices.

The design was thoroughly tested by the use of manual checking in an attempt to ensure that all requirements had been dealt with and that the design was consistent and feasible. This also ensured that coupling between program modules was loose and that the cohesion within each program was functional i.e. that each part of the program performs a clearly defined function. These attributes are considered to be important in good software engineering, particularly to aid maintainability.

IMPLEMENTATION AND VERIFICATION

The software was coded from the design in BASIC V for maintainability reasons, in so far as BASIC is the language most frequently used by, and familiar to, sports biomechanicians in the UK. Whilst not providing much of the data typing of e.g. Pascal, BASIC V does provide all of the control constructs and many of the parameter passing facilities of the later language. There are no great speed disadvantages in the use of BASIC V as opposed to Pascal or FORTRAN on the Archimedes 440.

The coding was undertaken by writing structured programs, i.e. ones that conform to the definition proposed by Linger et al. (1979) and are proper programs, the control flow in which can be represented by a flowgraph. This approach was adopted to facilitate verification and validation of the software. Implementation dependent features of the programs were confined, wherever possible, to a few modules.

The programs were firstly reviewed by detailed desk checking as recommended by Birrell and Ould (1988). This was done, in particular, to check for common faults such as syntax errors, wrongly nested conditional statements, incorrectly used variables and unitialised variables. After correction of any programming errors revealed by this method, each program module was then verified by a formal, or semi-formal, use of axiomatic semantics as outlined by Backhouse (1986).

This involves the specification, from the software design, using predicate calculus of pre- and post-conditions on code modules. The post-condition defines the state that the program is required to be in after the execution of that module, whilst the pre-condition defines the state that the program will be in before execution of the module. From the post-condition, program synthesis then works backward to establish the weakest pre-condition, i.e. a predicate describing the set of all initial states such that execution of the intermediate code begun in any one of these states is guaranteed to terminate in a state satisfying the post-condition. It is then necessary to show that the pre-condition subsumes this weakest pre-condition.

Such a thorough verification that software is built correctly is rarely reported for sports biomechanics applications.

SOFTWARE TESTING

Each program module was tested with specimen data sets obtained from equivalence partitioning of the input and output data space, along with boundary-value analysis to exercise the module on the boundaries of the equivalence classes. This black box approach, as recommended by Ould and Unwin (1986), allowed the development of representative and comprehensive test cases. It was supplemented by the use of structural (white box) testing using the independent paths coverage criterion. This ensures the execution during testing of a complete set of lineary independent paths through the code and requires a number of tests equal to the cyclomatic complexity of the module.

This testing procedure was then repeated for each complete program within the overall package. The purpose of testing in this way, which concurs with the recommendations of Myers (1979) was to ensure as far as reasonably possible that the correct product was built, i.e. the validation of the software, a different requirement from that addressed by verification.

USE OF THE PACKAGE

After this thorough verification and validation of the software, the package is being used in several studies of sports movements, e.g. an electromyographic/video investigation of wing kayak paddling techniques, cinematographic analysis of fast bowling in cricket and force platform analyses of jumping movements. Fuller details of the results of some of these studies will be reported elsewhere.

The software reported here in is sufficiently implementation independent to be easily adapted to other hardware, and should therefore be of interest to many researchers in the field. Whilst no claim is made that the software is fault free, the development methods used have produced a product which is far better engineered than much of the software used in sports biomechanics.

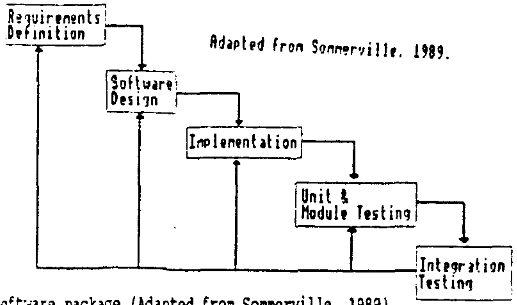

Figure 1: Lifecycle for the development of the software package (Adapted from Sommerville, 1989)

Figure 2. Data Flow Diagram for "Produce EMG Analysis" (level 2).

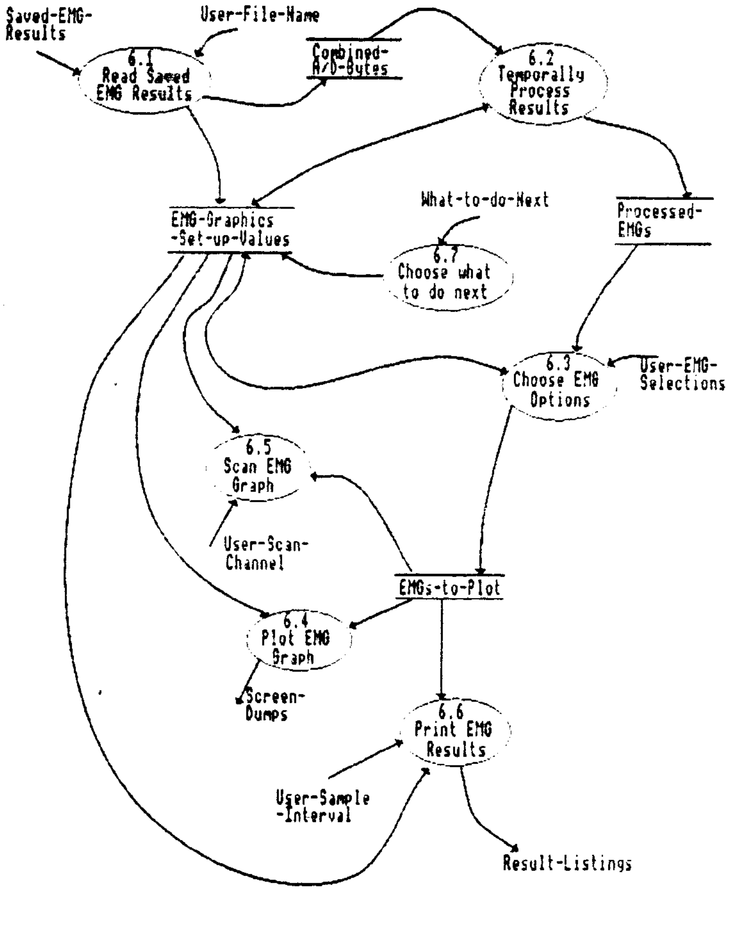

Figure 2: Data Flow Diagram for "Produce EMG Analysis" (level 2)

Figure 3. Finite State Machine Representation of Control Flow.

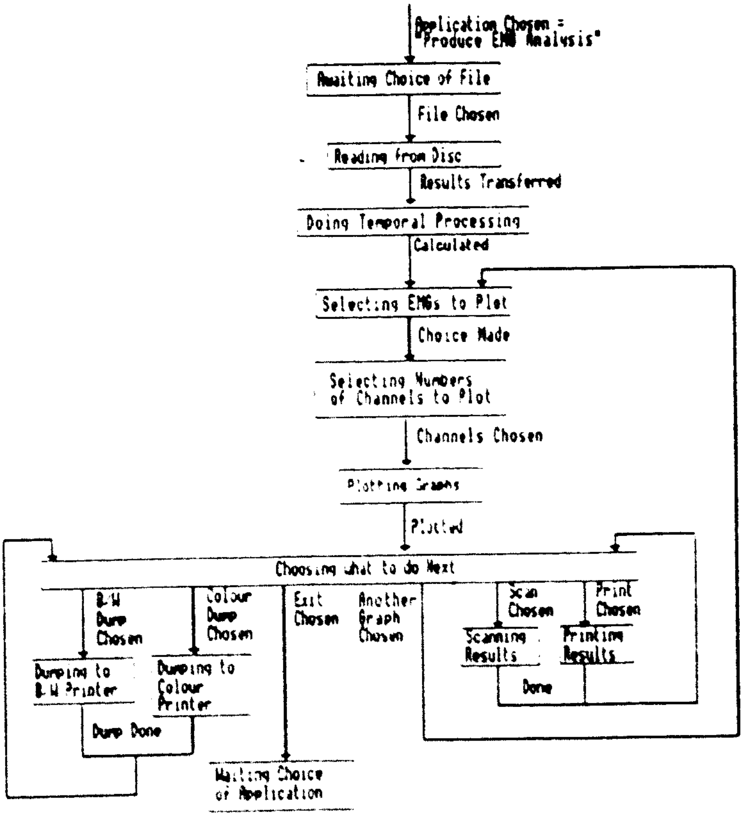

Figure 3: Finite State Machine Representation of Control Flow

REFERENCES

BACKHOUSE, R.C.: (1986) Program Construction and Verification. Englewood Cliffs: Prentice Hall.

BARTLETT, R.M.: (1989) Biomechanical Evaluation of the Elite Athlete. Leeds : British Association of Sports Sciences.

BIRRELL, N.D. and OULD,M.A.: (1988) A Practical Handbook for Software Development. Cambridge: Cambridge University Press.

CONSTANTINE, L.L. and YOURDON, E.: (1979) Structured Design. Englewood Cliffs: Prentice-Hall.

DAINTY, D.A. and NORMAN, R.W. (Eds.) (1988) Standardizing Biomechanical Testing in Sport. Champaign : Human Kinetics.

ISEK (1980) Units, Terms and Standards in the Reporting of EMG Research. USA : International Society of Electrophysiological Kinesiology.

KARARA, H.M.: (1980) Non-metric cameras. In: Atkinson, K.B. (Ed.) Developments in Close Range Photogrammetry, pp. 63-80. London: Applied Science Publishers.

LINGER, R.C., MILLS, H.D. and WITT, B.I.: (1979) Structured Programming : Theory and Practice. Reading, Mass.: Addison Wesley.

MYERS, G.J.: (1979) The Art of Software Testing. New York : Wiley.

OULD, M.A. and UNWIN, C.: (1986) Testing in Software Development. Cambridge : Cambridge University Press.

SOMMERVILLE, I.: (1989) Software Engineering. Wokingham: Addison Wesley.