The Development of DRUM: A Software Tool for Video-assisted Usability Evaluation

Miles Macleod and Ralph Rengger

National Physical Laboratory
DITC HCI Group
Teddington, Middlesex, TW11 0LW, UK

Tel: 081-943 6097 (+44 81 943 6097)
Fax: 081-977 7091 (+44 81 977 7091)
Email: miles@hci.npl.co.uk; Miles.Macleod@eurokom.ie

The development is reported of a practical software tool which supports video-assisted observational evaluation of usability. The Diagnostic Recorder for Usability Measurement (DRUM) helps evaluators to organise and analyse user-based evaluations, and to deliver measures and diagnostic data. This paper reports DRUM's rationale, theoretical background, requirements capture and collaborative iterative development. It outlines DRUM's functionality and manner of use. DRUM runs on Apple® Macintosh™, drives a range of video machines, and supports management of evaluation data, task analysis, video mark-up and logging (with find and replay of logged events), analysis of logged data and calculation of metrics.

Keywords: HCI, Usability, Observational evaluation, Tools, Video protocol analysis, Usability engineering, Usability metrics.

1. Introduction

There is a widely recognised need for cost-effective usability engineering and evaluation (Karat, 1992). This paper reports the development of a software tool which provides practical support for observational evaluation of usability, and can reduce video protocol analysis times to two or three hours per hour of video. DRUM, the Diagnostic Recorder for Usability Measurement, has been developed at NPL, initially within ESPRIT Project 5429 – MUSiC: Metrics for Usability Standards in Computing. DRUM supports quantitative evaluation, particularly the MUSiC Performance Measurement Method (Rengger et al., 1992), which is based on observation of task performance and analysis of how successfully people achieve task goals when using a system. DRUM also has wider applicability, assisting the generation and delivery of diagnostic feedback to a product's designers concerning usability defects.

DRUM, now in Version 2.0, has been developed iteratively since 1990, to meet the identified needs of usability evaluation. Development has involved close co-operation between HCI specialists, human factors professionals and software engineers, in collaboration with industrial users. DRUM supports the management of data at the different stages of observational evaluation, from selection of users to output of results. It provides

facilities for editing and browsing task analyses. It gives computer assistance for video control and the creation of interaction logs, with find and video replay of any logged event. It supports analysis of data, and calculation of metrics. DRUM has a graphical user interface, on-line context-sensitive and hypertext help, and a comprehensive user manual. It runs on Apple Macintosh hardware, and drives a variety of video machines.

Video recording offers considerable advantages for usability evaluation. Video clips of end-users working with a system can provide persuasive evidence for designers and developers of the usability of their system, and of specific problems. Video clips alone, though, are insufficient for usability evaluation; some kind of analysis is required. The first step in analysis, typically, is to produce a time-stamped log recording observed events. However, that raw interaction log will contain too much low-level detail to be of use to designers and managers. If what has been observed and logged can be analysed to give valid measures of usability, the evidence becomes convincing. DRUM facilitates such analysis, and provides assistance throughout the process of observational usability evaluation. Video logging and analysis has previously been very time-consuming, with expected analysis times of ten hours per hour of video. Using DRUM, this can be reduced two or three hours for each hour of video, depending upon the nature and level of detail of the analysis.

2. Theoretical Background

2.1. Methods for Evaluating Usability

Methods for evaluating usability can be categorised according to a number of different criteria (see, for example, Howard and Murray, 1987; Maguire and Sweeney, 1989; Whitefield, Wilson, & Dowell, 1991). A recent overview of approaches to evaluation is provided by Macleod (1992). Methods giving reasonably rich data about usability fall into three principal categories:

- Expert methods, based on expert judgement about a system or design (and hence dependent on available expertise).
- Theoretical methods, based on models of the user and system, and how they interact.
- User-based methods, where usability data are gained as a result of people using systems or prototypes. User-based methods divide broadly into survey methods, which give a picture of users' subjective views, and observational methods.

When a functioning system or prototype is available, user-based methods can provide arguably the most accurate and complete picture of usability. In any user-based evaluation it is desirable to ensure that the circumstances in which a prototype or system is evaluated match as accurately as possible the (intended) circumstances of system use. This includes characteristics of the users, the tasks they perform, and the organisational, physical and technical environments: in MUSiC these are collectively referred to as the Context. The MUSiC Context Guidelines Handbook (Maissel et al., 1991), provides guidance and support for this aspect of evaluation.

Observation of users, combined with an appropriate method for analysing what is observed, offers usability data of two kinds:

i. Diagnostic data - concerning usability problems and defects.

   A promising approach to diagnostic usability evaluation involves analysis of critical incidents and breakdowns, and is articulated by Wright and Monk (1989), drawing on the work of Winograd and Flores (1986). It is incorporated into a practical evaluation method by Wright, Monk and Carey (1990), who recommend the use of video recording.

ii. Performance data from quantitative evaluation; for example, the level of work goal achievement by various classes of user, performing representative tasks in particular environments.

    A method for video-assisted, user-based performance measurement has been developed within the MUSiC Project. The method gives measures of effectiveness and efficiency of work goal achievement, and measures of time spent unproductively by users, for example encountering problems and seeking help. It is documented in the MUSiC Performance Measurement Handbook (Rengger et al., 1992), and is being taken up by industry.

DRUM was developed to support the derivation of quantitative performance data and diagnostic data, and can be applied in a wide range of observational evaluation.

2.2. Recording and Analysing Data

Observational evaluation can be carried out most conveniently - from the evaluator's viewpoint - in a usability laboratory, where users can be provided with a simulation of the workplace, and sheltered from the observers by a two-way mirror (one way window). Alternatively, video and audio data and observational notes can be collected in the actual workplace, and the subsequent analysis conducted in a usability laboratory, or a suitably equipped office.

Usability laboratory analyses have in the past sometimes been described as being 'too focused on micro-issues' (Brooke, J., pers. comm., 1991). Such focus on low-level issues may arise from genuine academic interest in low-level events, or may be technology led: it is often easier to capture low-level information, for example by monitoring a system, than it is to record higher level data. However, when data are captured at the low level of keystrokes and mouse events, difficulties with data analysis can be encountered (Theaker et al., 1989; Maguire and Sweeney, 1989). Macleod (1989, 1990) describes a monitoring system which captures data at the rather higher level of user actions on objects such as menus and buttons; Kornbrot and Macleod (1990) report the spreadsheet-assisted analysis of such data. Hammontree et al. (1992) use tailored filters to sift low-level data.

The work reported here aimed explicitly to develop a practical and cost-beneficial tool, to facilitate observational usability evaluation and support usability analysis at a relatively high level. A principal goal in developing DRUM was to enable evaluators to deliver analyses of usability – providing measures and diagnostic data – at a level which can form the basis for evaluation reports which are comprehensible to people who make decisions about the development and purchase of software and systems.

The underlying principle is that evaluators observe, and - guided by a method - identify events of significance to the analysis. DRUM provides a simple means to record these in a time-stamped log. Evaluators can pre-define, at a chosen level of analysis, the types of events they wish to log, hierarchically organised if desired. Full control of the video is provided during logging of a tape. Once any event has been logged, it can be automatically located on the video, and reviewed. There is rapid access to previously created logs and other evaluation data files from its database. DRUM supports diagnostic evaluation (including the identification of evaluator-defined critical incidents) as well as measurement of usability.

3. Development Process

3.1. User Requirements Capture

During 1990, a number of 'brain-storming' requirements-capture sessions were held, at which experienced usability analysts from NPL and HUSAT Research Institute aired and discussed their requirements for a computer-aided video analysis tool. In addition, three products were identified which appeared to provide some of the functionalities envisaged as desirable for the proposed tool. These were: HIMS (developed by Manchester University in collaboration with the HUSAT Research Institute); EVA (produced by the Fraunhofer-Institute in Stuttgart); and the STL Usability Workbench (see Panel 1). The functionalities of all these products were reviewed, and members of the DRUM development team visited the organisations responsible for their development and were given demonstrations of the products.

Panel 1 : Products reviewed in defining DRUM requirements specification

HIMS: The Human Interface Monitoring System (Theaker et al., 1989; Maguire and Sweeney, 1989) is a research tool, constructed as part of a UK Alvey project at the University of Manchester Institute of Science and Technology. HIMS was developed on custom-made hardware for use with a domestic stereo VHS video-cassette recorder. It enables interactive sessions to be reconstructed from the system's point of view. Later versions capture display pixels instead of input data streams to avoid replay errors.

EVA II is a software tool for protocol analysis (Vossen, 1990) providing technical support for the analysis and evaluation of human behaviour. It runs on an Apple Macintosh II coupled with a Sony VO 7000 series video recorder. The software of the protocol system is implemented in HyperCard 1.2 and offers the following functions: construction of a scheme of behavioural events for an arbitrary task domain; application of the event classification scheme; browsing through the audio-visual protocols and modification on the basis of an event classification scheme; documentation of the resulting protocols in a form suitable for subsequent statistical analysis.

STL Usability Evaluation Workbench (Laws and Walsh, 1990) was developed on a Sun 3/60 workstation interfaced to a pair of Sony VO-7630 U-Matic VCRs and locally networked to an ICL mainframe computer. The Sun runs and controls the windows software but relies on the ICL mainframe for access to information on subjects and evaluation sessions held in a relational database. An emphasis is placed on experimental design.
No specification data were released to NPL, but some of the functions available on the Workbench are: control and display in a windows environment; a trial management phase, allowing markers to be set up in real time, video to be captured and scored and the data processed; VCR control with variable speed, search, freeze etc., controlled by screen menus, buttons and sliders; event logging, timing and description (annotation), use of behavioural categories; data processing and statistical analysis, intelligent searches, production of histograms, pie charts, hard copy etc.; report presentation; preparation of evaluation report videos.

We should like to acknowledge the influence these systems had on the identification of potential user requirements for DRUM, and thank the organisations for their co-operation.
Analysis of the results of the requirements capture sessions and the product reviews resulted in an initial requirements 'wish list' of functions for DRUM (see Panel 2).

Panel 2: Requirements specification 'wish list'

GENERAL

- Upward compatibility, with tool and data separation
- Data management facilities
- Sideways compatibility - environment common to Mac and PC platforms
- Modular construction, to allow for modification and expansion
- GUI with ergonomic screen layout - single display area preferred
- Easy navigation around modules and functions, with parallel multi-window displays of the logs and the analytical results
- Event locations and durations displayed and referenced in minutes and seconds from the start of the task
- On-line help, in support of paper-based manuals

SPECIFIC

Evaluation Manager
- Navigation by task, subject, analyst etc., through a database of evaluation data
- Selection of appropriate sessions, records and logs
- Support for manual datafile management in operating system environment

Record and Log Planner
- Setting up event monitoring and analytical schemes
- Setting up conditions for automatic event flagging

Record and Log Manager
- Support for real-time and post logging of events in the record
- Support for analysis and editing of the log
- Display and reference of event locations and duration by time
- Support for analogue and digital time monitoring
- Support for video monitoring
- Parallel multi-window displays of the logs and the analytical results

Session Recorder
- Synchronous recording of video, audio, analogue and digital data

Record Logger
- Synchronous playback of video, audio, analogue and digital data
- Logging of instantaneous events
- Logging of independent, exclusive and chained interval events
- Continuous availability of replay controls
- Variable replay speed control
- Compatibility between replay controls
- Safe browse mode available

Log Processor
- Statistical analysis and display available
- Automatic on-line derivation of the performance metrics from the logs
- Various print options and functions available
- Support for the export of the logged data and text
- Report generation

The individual requirements were prioritised, and the list provided the basis for the formal DRUM user requirements specification (Collins, 1991, following the BS 6719:1986 guidance on specifying user requirements for computer-based systems). During the development of DRUM, where individual requirements were found to conflict, some trade-offs were necessary in the degree to which they were met. Inevitably some limitations were also imposed by implementation constraints. The great majority of the requirements written into the specification have been met, as have further detailed user requirements identified in the course of iterative prototyping.

3.2. Development Method and Environment

The user requirements indicated that DRUM would be a tool with complex functionality, yet a fundamental prerequisite was that it should itself provide a high a level of usability, and enable effective use with minimal training. Experience of evaluating software produced using a traditional waterfall (or 'lemming') model of system development suggested that a rich prototyping approach was essential to tailor the evolving implementation more precisely to users' needs and capabilities. The order of programming complexity of DRUM's implementation is indicated by its 14,000 lines of code (HyperTalk, excluding comments); it was not possible at the specification stage to predict users' final preferences for many of the alternative detailed design solutions.

The adoption of a prototyping approach, with small and large iteration loops, made a development environment running interpreted code advantageous. A widely voiced view is that a prototype should at some point be re-implemented in a more efficient form. This however requires considerable resources, and effectively freezes development at that point, unless effort is available for further re-implementations. To gain maximum benefit from available resources, it was therefore decided that the prototyping environment would also be the delivery environment, provided it was capable of delivering the required level of performance.

3.3 From Requirements to Implementation

Initially it had been planned to develop versions of DRUM to run on Apple Macintosh and IBM PC platforms. PLUS™ was investigated: however, the functionality and speed of the available version of this were found not to meet our requirements. As no other suitable cross-platform solution was identified, development continued on Apple Macintosh. SuperCard™, which offered multiple windows and colour, was found not to meet our performance requirements. The functionality of HyperCard™ 1.2 was inadequate for our purposes (lacking re-sizeable windows, groupable text, and background fields displaying shared text). HyperCard 2 was chosen since it offered adequate functionality, and performance was found to be more than acceptable, particularly on higher powered machines. Care was taken throughout the development of DRUM to achieve acceptably fast execution of code, using comparative speed tests of algorithms. Where of advantage, key routines have been compiled.

An overall aim was that the implemented DRUM should meet users' needs in an effective, efficient and satisfying manner. To this end, a number of guiding principles were followed, which underlie good graphical interface design. A clear expression of a set of such guiding principles is provided by Apple® Human Interface Guidelines: The Apple Desktop Interface (Apple Computer, 1987), which also gives more detailed style guidance for Macintosh look and feel. In developing DRUM, we also drew on our own experience in developing and evaluating graphically interfaced systems. There were some compromises to be made between the Apple Desktop Interface style, and the HyperCard Human Interface Style: for example, whether single or double clicks are required to actuate or open an object. The card metaphor was avoided, but navigation through DRUM's modules is achieved by single clicks on buttons.

3.4. Video Control and Event Logging

The user requirements included, as a high priority item, support for real-time and retrospective logging of events. It is worth here considering briefly the rationale and implications of such a requirement. Some existing logging systems limit the evaluator to marking up event logs in real-time, and provide no direct contact between the logging

program and the video player. Logging which is limited to real-time introduces temporal and observational inaccuracies. It relies on rapid and accurate marking-up of the log, while the observer's attention is divided between the observation of the interaction and the control of the logging. It also does nothing to facilitate the more considered analysis of critical incidents. Retrospective logging usually involves pausing and reviewing, and therefore is only really practicable when the logging software can be used to drive the video player. During refinement of the requirements, it was considered essential that DRUM should support both real-time and retrospective logging of an extendible set of user-defined events as well as standard events; that the log should itself provide a means of controlling the video (by point and click); and that it should be possible to annotate events recorded in the log with comments, both in real time and retrospectively.

An aspect of video control essential to DRUM's user requirements is search, which enables the rapid location of previously-logged events. To be performed accurately, this requires video equipment which can register frame codes or time codes on the tape. This code can be vertically integrated in the video signal (VITC); a cheaper alternative is horizontal integration on an audio track (HITC). Communication between computer and video requires a computer interface board. DRUM communicates with this via the Mac serial port. Video machines have different command sets, and employ various different protocols for communicating commands. There are also different formats for communicating time and frame codes. HITC codes may not be displayed on the player. It was therefore necessary to implement different video-driver modules for DRUM, and to ensure that DRUM generically provides sufficient functionality, including display of time and frame codes. One technical problem is that video command protocols can require twelve or more bytes per command and may require inter-byte pauses. Displaying frame codes on a computer – during play or record – involves cyclical transmission of frame code requests, receipt of frame codes, translation and display. This can lead to unacceptable delays in the computer's response to concurrent user actions (such as logging). DRUM avoids this by interleaving checks for user actions into the cycle.

During the development of DRUM, various means of representing the event log were considered. An appealing idea was to produce a graphical trace (in real time or retrospectively), where the X axis represents time, and the states of one or more categories of event are represented on the Y axis, rather in the manner of a digitised multi-channel seismograph trace. This raises some interesting implementation questions, particularly where a log is built up by browsing back and forth through a recording to study specific events. The current version of DRUM presents the log in text form, rather than as a graphical trace. It meets the essential requirement that a log can be sorted into temporal order. For this DRUM employs a sort algorithm whereby, when two events share a common start time but have different end times, the longer (i.e. embracing, or parent) event is placed before the shorter.

4. DRUM's Modules and Functionality

4.1. Overview

DRUM is divided into modules, each supporting a different aspect of usability evaluation. User tests with early prototypes revealed that the nature and purpose of the modules (self-evident by this stage to the design team) were not immediately apparent to naive users, who were having difficulties forming an adequate overall conceptual model. To overcome this problem, an opening screen was designed which provides a high-level view of DRUM's

functionality (Figure 1). This gives direct access to each module, and introduces users to the navigation bar at the bottom of the screen, which is has a consistent screen location throughout DRUM. It also gives access to DRUM Help, and to the balloon help system which provides information about screen objects. Representations of the navigation bar, with relevant highlighting, are also used throughout the DRUM User Guide (a paper manual), to show at a glance which module of DRUM is being described at any point in the text.

4.2. DRUM Modules

The four modules provide support for:

- management of data through the various stages of observational evaluation
- task analysis, and representation of classes of events and usability problems
- video control and creation of interaction logs of evaluation sessions
- analysis of logged data and calculation of metrics

Figure 1: DRUM Overview Screen

4.2.1. DRUM Evaluation Manager

The Evaluation Manager provides an organised means for managing, manipulating and displaying data from the different stages of observational usability evaluation. DRUM uses ASCII files for data storage, allowing flexible compatibility with word processors, spreadsheets and statistics packages. The left hand area of the screen gives access to browsers for each type of data, and provides a graphical representation of the organisation of the different classes of data. Figure 2 shows the Log Browser, which gives access to log files from the current evaluation.

The Evaluation Manager browsers give access to data about:

- Subjects - the people being observed in evaluations
- Tasks - analytic schemes describing users' tasks and observable events
- Recording plans - technical arrangements for capturing raw data
- Evaluations - title and description. Further data relating to a given evaluation are grouped together, and can be accessed when that evaluation has been chosen from the Evaluations Browser.
- Video recordings of evaluation sessions
- Logs of observed user and system activities, created using DRUM
- Measures and metrics - calculated values for individual subjects and groups, derived from analysing logged task performance
- Reports of evaluation findings

Figure 2: DRUM Evaluation Manager, displaying the Log Browser

4.2.2. The DRUM Scheme Manager

Evaluators may wish to look out for many different kinds of event when studying a video record of an evaluation session. DRUM provides as standard a basic set of event types, identified in the MUSiC Performance Measurement Method as being of potential significance to the use of a system. Each event type is represented on screen as an event button. These buttons are used in the Recording Logger for logging observed instances of such events.

The DRUM Scheme Manager (Figure 3) enables evaluators to add and edit their own further event types (which may be sub-tasks or activities), and to create hierarchical descriptions of the tasks to be performed by subjects during evaluation sessions. Evaluator-defined events can be created, edited, renamed or deleted; parent events can be deleted complete with their children. DRUM supports hierarchical task analysis at up to four levels of decomposition. A browser provides simple navigation through the hierarchy.

Some events may occur at various stage of task performance (e.g. types of error): DRUM supports the creation of 'Pooled Event' buttons, which are accessible to the evaluator independently of the hierarchical event display.

Evaluators can attach a definition, categorisation and comments to each evaluator-defined event type; these are displayed in editable fields when the relevant event button is selected. Once a task-analytic scheme has been created, it can be stored in the DRUM Database for future use. Schemes can also be merged.

Figure 3: DRUM Scheme Manager

4.2.3. The DRUM Recording Logger

The DRUM Recording Logger (Figure 4) provides the functions needed to create and edit video-related logs of what is observed. It gives full control of the video in its two sub-modules: the Recorder assists log-creation in real-time, while recording on video; the Logger assists retrospective logging. In practice, the constraints of logging in real-time during an evaluation session limit what can be logged immediately. Typically, it is practical to mark-up key events, such as the start and end of tasks, and a relatively small set of types of incident. A general purpose Marker is provided for logging events of interest, which can then be easily located later for further analysis.

As a log is built up, comments can be added to logged instances of events. Log entries can retrospectively be edited, commented, sorted or deleted. Within the Recording Logger, logs can be stored in and retrieved from the database. They can be printed out, or loaded into the Metrics Processor for analysis.

The logger gives quite sophisticated control of the video recorder, including a variable speed shuttle (which pauses the tape on release, or can be locked in any position), and frame search. It provides automated location of any logged event on the video, by point and click at the event frame code in the log display. When an event start frame is clicked on, DRUM winds the video to that frame, and - if the event has a duration, rather than being instantaneous - offers the opportunity to view that event clip. The tape is paused at the end of the event.

Figure 4: DRUM Recording Logger

4.2.4. The DRUM Log Processor

The Log Processor (Figure 5) performs the calculation, from any log in the DRUM database, of performance measures and performance-based usability metrics (see Rengger et al. 1992), including:

- task time - total performance time for each task being studied (with a facility for subtracting times when the task is suspended);

- snag, help and search times - measures of the time users spend having problems, seeking help or unproductively hunting through a system;
- effectiveness - derived from measures of the quantity and quality of task output, this is a measure of how fully, and how well, users succeed in achieving their task goals when working with a system;
- efficiency - this relates effectiveness to the task time: it is a measure of the rate of producing the task output;
- relative efficiency - a measure of how efficiently a task is performed by a specific user or group of users, compared with experts (or with the same task on another system);
- productive period - the percentage of task time not spent in snag, help and search. This indicates how much users of a system spend their time working productively towards their task goals.

Results are displayed in numerical and graphical form, and can be saved in the database for grouping of data from different subjects, for further statistical analysis.

Figure 5: DRUM Log Processor

5. Help System and User Manual

DRUM is supported by a paper-based user manual (Macleod et al. 1992), and provides two kinds of on-line help:

i. The 'balloon help' system gives the user information about objects visible on screen. When balloon help is turned on, positioning the mouse pointer over an object brings up a 'speech balloon', giving a brief outline of the purpose and use of the object. This kind of help is most useful during initial exploration and learning.

ii. 'DRUM Help' (Figure 6), a small program which co-exists with DRUM, provides 114 screens of textual and pictorial explanation and guidance: it gives more procedural, task-related information. Clicking on any item in the Help index takes the user to the relevant screen. Where appropriate, pop-up fields provide additional information. There are hypertext links between terms in the text and other screens; each screen also gives an index of related topics.

Modal dialog boxes within DRUM also give direct access to contextually relevant information in DRUM Help, via a Help button.

Figure 6: Two views of DRUM Help

6. Applying DRUM

DRUM can provide support for a wide range of video-assisted observational evaluation. Organisations already using DRUM have applied it in support of in-house evaluation methods, as well as the specific MUSiC method. Guidance for the design, planning and conduct of quantitative observational evaluation of usability is provided by DRUM's companion tools:

• The MUSiC Context Guidelines Handbook (Maissel et al., 1991) provides firstly a simple method for describing key characteristics of the users, tasks and work-environments for which a system is designed - the 'Context of Use'. Using a questionnaire format, it then gives the evaluator a structured method for describing characteristics of the users, tasks and

environment in which the system is to be evaluated, and of documenting how accurately these match the intended context of use.

• The MUSiC Performance Measurement Handbook (Rengger et al., 1992) introduces and explains the method which DRUM was developed to support: it provides instructions for each stage of evaluation, from planning and design to interpretation of results. It includes a quick guide, and technical appendices on usability metrics, problem descriptions, and hierarchical task analysis.

7. The Usability of DRUM

A question frequently asked about tools which support usability evaluation is 'has it been used to evaluate its own usability?'; in the case of DRUM, the answer is 'yes'. An evaluation study of the use of DRUM 1.0 was conducted by an evaluator not directly concerned in the development of DRUM 1.0 (Cant, 1992). The study involved the performance of representative evaluation tasks – which had been identified in a context of use study – by seven usability professionals experienced with DRUM Version 1.0, working in typical usability laboratory environments. Their task output was evaluated, and video recordings analysed using DRUM. User satisfaction was measured with SUMI, the Software Usability Measurement Inventory, a fifty item standardised questionnaire developed within MUSiC (Porteous and Kirakowski, 1992; Kirakowski et al., 1993). With the caveat that the recommended number of users for a quantitative evaluation is ten or more, the study provided baseline measures of DRUM's usability for comparison with other tools supporting those tasks, and provided valuable diagnostic data which have been used in the development of DRUM Version 2.0. The results demonstrated that DRUM, even in Version 1.0, met the acceptance criteria laid down in the requirements specification.

8. Further Development of DRUM

Work is in progress to extend DRUM's support for diagnostic evaluation. The Diagnostic Processor will provide data about frequency, duration and location of evaluator-defined events (e.g. critical incidents, or different categories of error). Other planned enhancements include: extension of the log sorting functions, to enable grouping of related but temporally distant events; enhanced graphical displays; grouping of event clips; and further video driver modules.

Panel 3: DRUM - technical details

DRUM requires:
- Apple Macintosh II computer, with a 13" or larger monitor
- System 7 (or System 6.0.5 or later)
- HyperCard™ 2.0v2 or later, allocated at least 1.3 MB RAM

DRUM can at present drive the following video recorders:
- Sony U-Matic VO 7000 and 9000 series, with BKU 701 computer interface, plus Sony FCG-700 frame code generator
- Sony Hi-8 with Video Schaay computer interface
- Panasonic AG-7350 or AG-7355 SuperVHS, with AG-IA232TC RS-232 board & time code generator/reader

Acknowledgements

Thanks are due to all those who have contributed the development of DRUM, especially: Nigel Bevan, Miranda Blayney, Rosemary Bowden, Vicki Cant, Stephen J. Collins, Clare Davies, Andrew Dillon, Annie Drynan, Ian Hosking, Edo Houwing, Rachel Jenkinson, Karl London, Martin Maguire, Jonathan Maissel, Brian J. Scott, Marian Sweeney, Cathy Thomas, and Paulus Vossen.

This work was supported by the Commission of the European Communities and the UK Department of Trade and Industry.

References

Apple Computer (1987), "Human Interface Guidelines: The Apple Desktop Interface", Wokingham, England, Addison-Wesley.

Cant, V (1992), "Usability Measurement of Information Technology Systems", HUSAT Memo 583, HUSAT Research Institute, Loughborough, UK.

Collins, S J (1991), "DRUM; Specification of User Requirements", NPL.M1.TW3.1. ESPRIT Project 5429 - MUSiC, Restricted document, NPL, Teddington, UK.

Hammontree, M L, Hendrickson, J J, and Hensley, B W (1992), "Integrated Data Capture and Analysis Tools for Research and Testing on Graphical User Interfaces". In Proc. CHI'92, ACM Press, pp. 431-432.

Howard, S and Murray, D M (1987), "A Taxonomy of Evaluation Techniques for HCI". In: Bullinger, H-J and Shackel, B (Eds), Human-Computer Interaction - INTERACT'87, (Proceedings of 2nd IFIP Conference on HCI) Elsevier, Amsterdam. pp. 453-459.

Karat, C-M (1992), "Cost-benefit and Business Case Analysis of Usability Engineering". CHI'92 Tutorial Notes, ACM Press.

Kirakowski, J, Porteous, M and Corbett, M (1993), "How to Use the Software Usability Measurement Inventory: The User's View of Software Quality". Proceedings of European Conference on Software Quality, 3-6 November 1992, Madrid.

Kornbrot, D and Macleod, M (1990), "Monitoring and Analysis of Hypermedia Navigation". In Diaper et al. (Eds) Proceedings of INTERACT '90, 3rd IFIP Conference on HCI (Cambridge, UK, 27th - 31st August), Elsevier, pp. 401-406.

Laws, J and Walsh, P (1990), The STL Usability Workbench. BNR Europe, Harlow, UK.

Macleod, M (1989), "Direct Manipulation Prototype User Interface Monitoring". In Sutcliffe, A and Macaulay, L (Eds) People and Computers V (Proc of HCI'89 Conference, Nottingham, 5-8 September), Cambridge University Press, pp. 395-408.

Macleod, M (1990), "Tools for Monitoring and Analysing the Use of HyperMedia Courseware". In Oliveira, A (Ed.) Structures of Communication and Intelligent Help for Hypermedia Courseware, Proceedings of NATO Advanced Research Workshop, Espinho, Portugal (19-24 April), Springer-Verlag.

Macleod, M (1992), "An Introduction to Usability Evaluation". National Physical Laboratory, DITC, Teddington, UK.

Macleod, M, Drynan, A, and Blayney, M (1992), "DRUM User Guide". National Physical Laboratory, DITC, Teddington, UK.

Maissel, J, Dillon, A, Maguire, M, Rengger, R, and Sweeney, M (1991), "Context Guidelines Handbook". MUSiC Project Deliverable IF2.2.2, National Physical Laboratory, Teddington, UK.

Maguire, M and Sweeney, M (1989), "System Monitoring : Garbage Generator or Basis for Comprehensive Evaluation System". In Sutcliffe, A. & Macaulay, L. (Eds.), People and Computers V (Proc of HCI'89 Conference, Nottingham, 5-8 September), Cambridge University Press, pp. 375-394.

Porteous, M and Kirakowski, J (1992), "SUMI: The Software Usability Measurement Inventory". Human Factors Research Group, University College, Cork, Ireland.

Rengger, R., Macleod, M., Bowden, R., Bevan, N. and Blayney, M. (1992), "MUSiC Performance Measurement Handbook". National Physical Laboratory, DITC, Teddington, UK.

Theaker, C J, Phillips, R, Frost, T M E and Love, W R (1989), "HIMS: A Tool for HCI Evaluations". In Sutcliffe, A. & Macaulay, L. (Eds.), People and Computers V (Proceedings of HCI'89 Conference, Nottingham, 5-8 September), CUP, pp. 427-439.

Vossen, P. (1990). EVA II (Software Tool for Video Protocol Analysis), Fraunhofer-Institut für Arbeitswietschaft und Organisation, 7000 Stuttgart 80, Nobelstraße 12, Germany.

Whitefield, A, Wilson, F, and Dowell, J (1991), "A Framework for Human Factors Evaluation", Behaviour and Information Technology, 10(1), pp. 65-80.

Winograd, T and Flores, F (1986), "Understanding Computers and Cognition: A New Foundation for Design", Wokingham, UK, Addison Wesley.

Wright, P and Monk, A F (1989), "Evaluation for Design". In Sutcliffe, A. & Macaulay, L. (Eds.), People and Computers 5, Cambridge University Press, pp. 345-358.

Wright, P, Monk, A.F. and Carey, T. (1990), "Co-operative Evaluation: The York Manual", University of York, UK.