Building Bridges between Human Factors and Software Engineering

D. Bell[§], A. Gupta[§], H. Rozendaal[¥] & E. Spencer[‡]

[§]Philips Research Laboratories, Cross Oak Lane, Redhill, Surrey, UK, RH1 5HA
Tel: +44 1 293 815 000
Fax: +44 1 293 815 500
Email: {bell,gupta}@prl.philips.co.uk

[¥]Philips Industrial Electronics, Analytical X-Ray, Lelyweg, Almelo 7602 EA, The Netherlands
Tel: +31 546 839 421
Fax: +31 546 839 593
Email: wswrozendaal@amlie3.decnet.philips.nl

[‡]EDS, 1-3 Bartley Heath Business Park, Hook, Hampshire, UK, RG27 9XA
Tel: +44 1 256 742 458
Fax: +44 1 256 742 700

Abstract

Whilst there is much in the literature on methods for user interface design, reports on integration with software development are few. Communication amongst members in multi-disciplinary teams is vital, as is the scheduling of mutual reviews . In this paper, we describe our experience in using the Method for USability Engineering (MUSE), which addresses these issues, and identify the consequent software process improvement and the areas that necessitate further consideration. We believe our initial observations will be of interest to a wide audience in the domains of human factors and software engineering. This work is being conducted under the auspices of the CEC's ESSI (European Software & Systems Initiative) programme.

Keywords: user interface design methods, software engineering, process improvement.

1. Introduction

Users increasingly base their judgement of the functionality and quality of a software product on their perception of its interface. The identification of techniques that improve the interface are therefore of great interest to developers of software and other products alike. By addressing issues early in the design cycle, redesign and maintenance costs may be reduced. By improving the integration between human factors (HF) on the one hand and the software engineering (SE) and product development team on the other, good HF practice can be incorporated into products. However, three issues are frequently identified as negating these benefits:

1. multi-disciplinary teams make communication difficult because of the different vocabularies and concerns of the specialists involved (Grudin, 1992);
2. HF and SE staff use different tools and techniques;
3. managing the timely collaboration and exchanges between the HF and SE teams poses new problems for development managers.

1.1 **Objectives of the Project**

The problems identified above, have brought the authors together in a project where we each address our different application and process integration problems in our respective contexts, but use (and evaluate) a common interface design method, namely MUSE. The project, which is still ongoing, is centred on evaluating the suitability of MUSE to support user interface design in conjunction with different SE methods. This cost-benefit evaluation will be achieved by the use of cross-project metrics applicable to the method. Alternative views of an integrated cycle are given in (Brown, 1994), (Bass & Coutaz, 1991), (Sutcliffe & McDermott, 1991), (Catterall, 1991), (Macaulay et al., 1990), (Curtis & Hefley, 1994), (Hix & Hartson, 1993) and (Wasserman, 1986).

This paper is organised as follows. In Section 2 we introduce the MUSE method; only a cursory description is provided and the interested reader is referred to the literature (Lim & Long, 1994; Lim & Long, 1992). This is followed by a description of our initial findings on using MUSE and on the metrics we are using.

2. **The MUSE Method**

MUSE defines the user interface design process by specifying the (intermediate) products to be generated and the procedures to be followed. Many products are based on the structured diagram notation of the Jackson System Development method (Jackson, 1983). Each product is supported by a table containing comments and design speculations. It should be noted that presently MUSE has no software support.

MUSE consists of three phases. The first, i.e. the Information Elicitation and Analysis phase, involves collecting and analysing information (using techniques such as task analysis) concerning the existing tasks of users and the positive and negative features of the systems that they currently use. The information is summarised in a device-independent task model of the existing systems; the system to be designed is modelled at a similar level based on the Statement of Requirements. In the second, i.e. Design Synthesis phase, the HF requirements are established and the semantics of the task domain recorded. A conceptual design for the target system is produced; this is based on the device-independent task models and informed by the information collected about the task domain and the HF requirements. This is then split to separate those sub-tasks that are to be performed using the system under development from those that are performed using other devices. In the final, i.e. Design Specification phase, a detailed and device-dependent specification of the user interface is produced; this includes descriptions of the screens, widgets and the interactions. The design is then assessed and refined in an iterative fashion.

Initial Assessment of the Value of MUSE

MUSE represents an approach which is highly focused on the design of user interfaces. It lays little claim to support other areas of HF, for example with respect to system design. In fact MUSE's principal claim is that it offers a process that leads to the production of an HCI specification and it defines the form of this specification and the steps required to generate it. It does not seek to teach people how to design or to restrict the input of novel ideas. What it does attempt to do is provide a structure within which such ideas and information can be documented and used; it prompts for a conscious process of design and provides a record of the design process.

At present, we are more than half-way through the MUSE process. The following comments are made with respect to the three issues identified at the top of this paper.

2.1 **Multi-disciplinary teams make communication difficult because of the different vocabularies and concerns of the specialists involved**

Lessons Learnt - Benefits

The notation utilised within MUSE - the structured diagram notation taken from the Jackson System Development method (with additions such as tables and a semantic net) - has proven to be easily understood by SE, designers, and HF experts. Remarkably, we have also found it to be easily understood by the application domain experts, i.e. task performers, to whom we sent the task descriptions for verification. This is of great benefit for it means that a design team which is both multi-disciplinary and geographically distributed will be able to discuss issues related to analysis or early design with reduced ambiguity. It should be noted though that our task performers all have a scientific/engineering background. However, the notation is inherently understandable by a wider audience.

Lessons Learnt - Shortfalls

Although the notation has been found to be easily readable, we are finding problems with the notation that relate to representing specific design ideas. Essentially it is a reliability problem: different analysts produce different representations of the same task or HCI design constructs. This reliability issue also brings into question the apparent gain of ease of readability. It is possible that the notation could be interpreted differently, leading to different conclusions being drawn, or at the extreme different designs.

2.2 **HF and SE staff use different tools and techniques**

Lessons Learnt - Benefits

The use of a common notation for the majority of the MUSE products offers two benefits. Firstly the notation provides a vehicle to translate the results of other methods into a form which is used by both the HF and SE teams. In many instances there is no need to actually translate. The notation may be adopted directly to document the results of the analysis technique.

Secondly the use of a common notation across all stages of the project provides a consistent view of the evolving design. The results of specific techniques are 'added' into the evolving design at each stage.
Lessons Learnt - Shortfalls

The notations available do not provide the means to record and convey the breadth of knowledge gained from specific techniques such as task analyses, for example the language of the application domain, differences amongst users etc. In addition, it is difficult to link domain information, such as object descriptions and attributes, to the task descriptions. The semantic net description in MUSE addresses this but is only successful to a limited extent due to a fundamental problem, namely, the gulf in the abstractions and scope of task analysis on the one hand and SE driven domain analysis on the other.

2.3 **Managing the timely collaboration and exchanges poses new problems for development managers**

Lessons Learnt - Benefits

The MUSE method supports the integration of H F and S E activities by specifying mandatory "handshakes" (i.e. cross-checking) points between the two activities.

The first of these handshakes is undertaken to ensure that the two activities have a common understanding of the functionality to be provided by the system. The members of the HF stream can check that the functionality they feel is required can be provided, and the SE stream can ensure that they are not going to provide superfluous functionality. This handshake ensures that the efforts of the two streams does not result in incompatible designs. The second handshake is undertaken at the end of the MUSE activity to ensure that the "look and feel" is implementable.

In addition, the particular needs of a project or SE process, for example, may dictate ad hoc or discretionary handshakes to be defined and undertaken. These can, for example, be used to ensure common understanding of the initial project brief, or to ensure that the two streams are using the same terms to describe the application domain and share a common interpretation.

Lessons Learnt - Shortfalls

UI developers are often asked to explain the reasoning for the implementation of UI features. The traceability of a screen object or of an interaction through to its origin in a design document such as a task description, is essential to enable such questions to be answered. Requirements need to be tested in relation to the design as it evolves at each stage of the design; there is no support for this.

In addition, traceability throughout the design process is vital in evaluating, at each stage, how successful the current solution is at meeting the requirements. Many of the requirements will have a human component, e.g. the undertaking of specific task will contribute to overall system performance. Testing that the system meets its requirements needs to take account of all aspect of the design.

3. Metrics

The metrics we shall be applying to evaluate MUSE, can be divided into those that measure the extent to which it has been applied correctly and those that measure the costs and benefits of producing individual products. Cost is to be measured and calculated from a subset of the primitive measures shown in Figure 1. These include recording the size of each document, the effort and duration expended in its production and the effort involved in finding and rectifying defects. Benefits of producing individual documents will be recorded firstly by use of a subjective 'usefulness' measure, and secondly, by recording the frequency of use of each product. In addition, any limitations of the MUSE notation will be recorded to judge its richness in being able adequately to document all elicited information.

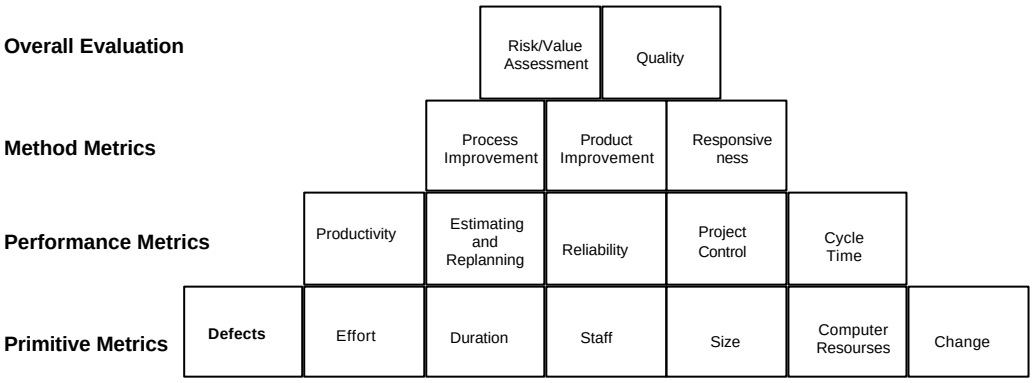

Figure 1: Primitive and Calculated Metrics

The cost benefit evaluation of MUSE uses only those highlighted

4. Conclusions

Whilst MUSE offers an improvement on the software methods in use by the project partners, it cannot guarantee a good design. Our evaluation is based upon the integration of user interface issues early in the software life cycle and we have identified requirements which will entail revisions to MUSE, or contribute to the development of a new approach. Our evaluation is not yet complete; at the time of writing the partners are well into the Design Synthesis phase. We will have completed our evaluation by end-July when the project is scheduled to terminate.

Acknowledgements

The Ergonomics and HCI Unit, University College London, are sub-contractors to this ESSI project; we are grateful to Prof. John Long and James Middlemass for the training given to us in using the MUSE method, for the many discussions we have had, and for contributing to Section 2 of this paper.

References

L. Bass & J. Coutaz (1991), *Developing Software for the User Interface*, Addison-Wesley.

D. Browne (1994), *STUDIO Structured User-Interface Design for Interaction Optimisation*, Prentice Hall.

B. Catterall (1991). *Three approaches to the input of human factors in IT systems design: DIADEM, The HUFIT Toolset and the MOD/DTI Human Factors Guidelines*, Behaviour & Information Technology, 10(5):359-371.

B. Curtis and B. Hefley (1994). *A WIMP No More - The Maturing of User Interface Engineering*, ACM Interactions, Jan. 1994, pp.22-34.

J. Grudin, J (1992). *Utility and Usability: research issues and development contexts.* Interacting with Computers, (4)2:209-217.

D. Hix and H. R. Hartson (1993). *Developing User Interfaces: Ensuring Usability Through Product and Process*, Wiley, New York.

K. Y.Lim and J. Long (1992). *A method for recruiting methods: Facilitating human factors input to systems design.* Proceedings CHI'92, pp.549-556.

K. Y. Lim and J. Long (1994). *The MUSE Method for Usability Engineering*, Cambridge University Press.

M. Jackson (1983). *Systems Development*, Prentice Hall.

L. Macaulay, C. Fowler, M. Kirby and A. Hutt (1990). *USTM: A New Approach to Requirements Specification*, Interacting with Computers, (2)1:92-118.

A. G. Sutcliffe and M. McDermott (1991). *Integrating methods of human-computer interface design with structured systems development*, Int. J. of Man-Machine Studies, (34):631-655.

A. Wasserman, P. Pircher, D. Shewmake and M. Kersten (1986). *Developing Interactive Information Systems with the User Software Engineering Methodology*, in Readings in HCI, Baecker & Buxton [eds], pp.508-527, Morgan Kaufman.