Enhancement of the Software Process: Human Factors as an Integral Component

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

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

[§]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

Abstract

Human Factors can (and does) contribute to all aspects of the development and implementation of effective software systems. Its involvement in identifying the needs of users with respect to the design of the Human-Computer Interface (HCI) frequently represents the most visual element. Whilst there is much in the literature on methods for user interface design, reports on its integration with software development are few. In achieving integration, 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 some areas that require further work. We believe our observations will be of interest to a wide audience in the domains of human factors and software engineering. This work was conducted under the auspices of the CEC's ESSI (European Software and Systems Initiative) programme.

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

1. INTRODUCTION

1.1 THE PROBLEM OF INTEGRATION

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 user issues early in the design cycle, redesign and maintenance costs may be reduced.

The application of human factors expertise to support all phases of the life cycle provides methods and techniques through which the user viewpoint is able to contribute to, and impact on, the design of the system. A number of human factors techniques and tools exist to support the design of an effective interface. What is lacking is guidance on how to make use of the information from these techniques to support the software process and when it is appropriate for this information to be provided.

By improving the integration between human factors on the one hand and the software engineering and product development team on the other, good human factors practice can be incorporated into products. However, a number of issues are frequently identified as negating these benefits:

- Multi-disciplinary teams make *communication* difficult because of the different vocabularies and concerns of the specialists involved (Grudin, 1992);
- The resulting HCI specification is difficult to implement and does not *specify* the dynamics of the interface;
- Even where the HCI is perceived to be good the system often proves to be *ineffective*, is used improperly or not used at all.

1.2 OBJECTIVES OF THE PROJECT

The problems identified above, 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 the Method for USability Engineering (MUSE) (Lim & Long, 1994; Lim & Long, 1992). The project is centred on evaluating the suitability of MUSE to support user interface design in conjunction with different software engineering methods. A cost-benefit evaluation has been undertaken by the use of cross-project metrics applicable to the method. Related work in this area is described in (Brown, 1994), (Bass & Coutaz, 1991), (Sutcliffe & McDermott, 1991), (Catterall, 1991), (Macaulay et al., 1990), (Curtis & Hefley, 1994), (Hix & Hartson, 1993) and (Wasserman, 1986).

The remainder of 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). After undergoing training in the method, directly from the developers of MUSE, we applied the method on independent projects. This is followed by an overview of our principal conclusions.

2. THE MUSE METHOD

MUSE defines the user interface design process by specifying the (intermediate) products[1] to be generated and the procedures to be followed. These 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 the following three phases.

Information Elicitation and Analysis Phase: involves collecting and analysing device dependant 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 and of other selected systems. The information is summarised in a device-independent (i.e. high level) task model of the existing systems; the system to be designed is modelled at a similar level based on the Statement of Requirements, i.e. the Project Brief.

Design Synthesis Phase: the human factors 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 task models produced earlier, and informed by the information collected about the task domain and the human factors requirements. This is then split to separate those sub-tasks that are to be performed

[1] All MUSE products represent design documents and should not be confused with software products.

using the system under development from those that are performed off-line.

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.

MUSE represents an approach which is highly focused on the design of user interfaces. It lays little claim to support other areas of human factors, for example with respect to job 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 promotes an explicit design process and provides a record of this process.

3. ASSESSMENT OF THE VALUE OF MUSE

The following section discusses our principal conclusions with respect to each of the three issues identified in Section 1.1 above.

3.1 IN AIDING COMMUNICATION

Handshaking

The MUSE method supports communication between human factors and software engineering activities by specifying mandatory "handshakes" (i.e. cross-checks) at points between the two groups of activities. These handshakes represent bridges between the human factors and software teams.

Whether the project is organised as one integrated team or as separate teams, human factors and software engineering are separate disciplines in their own right. There will be communication problems which handshakes address.

The first of these handshakes is undertaken to ensure that the two activities have a common view of the functionality to be provided by the system. The members of the human factors stream can check that the functionality they feel is required can be provided, and the software engineering stream can ensure that such functionality can be implemented or is not going to be superfluous to the requirements. 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 software engineering 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.

The concept of handshakes allows for planned and controlled exchanges of information. The handshakes define the products and therefore the scope of the information to be considered jointly by human factors and software analysts. However, given the definition of a product that is of such importance to all parties then it could be argued that this should be produced as part of an integrated programme of human factors and software engineering activities. The concept of handshaking may provide a bridge, but it does emphasis the divide.

Language

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 software engineers, designers, and human factors 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.

Although the notation has been found to be easily understood, some problems were discovered with the notation that related to representing specific design ideas. Essentially it is a reliability problem: different analysts were found to produce different representations of the same task or HCI design constructs. If the same task or design can be represented differently then it is possible that a representation could be interpreted differently, leading to different conclusions being drawn, or at the extreme different designs being developed. This reliability issue serves to qualify (but not contradict) the benefits noted above..

Design Rationale

Developers of the interface are often asked to explain the reasoning for the implementation of user interface 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. MUSE's analytical approach allows for the design decisions to be recorded and the reasoning to be expressed with respect to the goals the user is trying to achieve.

Domain Knowledge

Domain knowledge is vital in the design of the HCI. It provides the context within which the system will operate. Specifically the jobs that the users will perform need to be reflected and supported by the HCI. Lack of a user/task focus during the design stage is a major factor leading to inefficient or unusable systems.

MUSE offers a task viewpoint on the design problem. This viewpoint is particularly strong because it extends throughout the design process. The use of a task-based representation of the design problem and more importantly the sharing of this view with the software engineering team offers a mechanism to allow all members of the team to understand the job for which the system is being developed. MUSE exclusively provides a task view. It does not directly consider the wider organisational and operational environment context.

Human factors consultants are frequently seen (and see themselves) as *the* channel of communication between the users and the software designers. Early task-based analysis methods are employed, such as task analysis. However, the transfer of this information to the design team frequently only occurs in an *ad hoc* or unplanned way. A number of factors contribute to the problem:

- The volume of information collected;

  It is often not seen as appropriate for all members of the team to spend time reading the wealth of information.

- The form of that information;

- Early information collection may not be focused on specific aspects of the system with which the designers are concerned. Additionally the structure of the information does not assist easy assimilation.

- Senior designers are also in discussion with user representatives.

  Members of the software team will be in discussion with user representatives and often feel that they have a good overview of user's problems. However, frequently they have been talking about distinct system implementation problems and have not covered the breadth of user issues.

MUSE offers the advantage of representing the collected domain knowledge in a design document. The design document will hold more weight and receive a more effective review by the software team. The method also provides for pulling this domain knowledge through the design process.

3.2 IN PROVIDING AN HCI SPECIFICATION

Representation

MUSE defines the HCI specification as a series of related products. These products present:

- Pictorial representations of the screens;

- The dialogue design representing the paths between the various screens;

- The dialogue design for each screen;

- A description of each component on each screen;

- The definition of error messages that are presented as a result of a false action or input on each screen.

This set of related products provides for a comprehensive specification. It provides both a static and dynamic view of the operation of the HCI. Although not unique in its scope, the task based representation of the dialogue allows for communication to both software engineers and user representatives.

Extent of MUSE in the Life cycle

Traceability of the Requirements

MUSE offers no explicit mechanism to support the traceability of requirements through each of the products. The traceability of requirements is of fundamental importance in ensuring that the design completely meets the identified needs of the user.

Traceability can be supported by use of the tables that support each of the MUSE products, e.g. another column can be added to all product tables to cross-reference the requirements. However, there is a distinct overhead in maintaining these tables when, for example, the structured diagram is restructured. Additionally, matching the requirement to the nodes in the diagram and rows in the table can be difficult. A requirement does not always correspond directly to set of nodes in a given branch. Similarly, a set of nodes may address more than one requirement.

Evaluation and Acceptance

The HCI specification that results from the MUSE process should be the subject of evaluation and ultimately acceptance testing. Both of these elements occur outside of the context of MUSE. Additionally the evolving design should be tested at each stage. Testing should be performed against the requirements. The type of test will vary over the course of the design, for example:

- Analytical tests, i.e. paper based;
- Check lists;
- Modelling tests;
- Prototyping tests;
- Equipment trials.

The purpose of the tests will also vary, for example:

- Completeness of the design;
- Comparing design options;
- Performance testing;
- Usability testing;
- Acceptance testing.

Usability evaluation is an area where human factors experts would traditionally expect considerable involvement. Usability testing should be brought within MUSE. Currently there is no concept of proving the design from this standpoint although the method does encourage the importing of usability objectives. Unfortunately the method only uses these objectives as guidelines to support the development of later phase products. We believe that usability objectives have greater impact on the design where they are identified early in the analytical process and included in the statement of requirements. By including them in the statement of requirements they will be the subject of testing at the acceptance stage.

Acceptance is the term used here to refer to the formal series of tests performed on the final product, i.e. to verify to the customer that the system does what was originally asked. Acceptance testing is a distinct process in the system life cycle. To date, little user related testing has been performed at this stage even where human factors activities regularly contribute to the design process. This is due to a variety of factors:

- The lack of user requirements in the requirements specification
- The difficulty of formulating user requirements such that they are testable;
- The problems of testing the requirements.

The above factors are all inter-related and have provided a circular argument which has encouraged the view of human factors as an 'add-on' or a 'black art'. What is needed is a frameworks that provides for human factors activities to be integrated directly into the design and acceptance process. This must include the formulation of testable requirements. Such a framework could extend the MUSE process over the entirety of the system life cycle.

Once user requirements are written into the requirements specification, the problem of 'too little too late' that MUSE sets out to address is removed. When there are user requirements that the system will be tested against then the need for human factors inputs becomes mandatory.

3.3 IN DELIVERING AN EFFECTIVE SYSTEM

The Scope of MUSE

MUSE is directed explicitly at the design of the HCI. It defines a relationship between this process and that of software engineering. The relationship of MUSE to other human factors areas or areas where human factors impacts is discussed as part of the documentation of the method. However, explicit links between these areas are not provided by MUSE.

A set of distinct technical areas can be used to explore the limited scope of MUSE with respect to human factors. These technical areas are:

Human Computer Interaction: All aspects of the design and specification of the HCI for the system.

Equipment: All aspects of definition or design of hardware and software for the system.

User: All aspect of the tasks to be performed by the various users of the system, including the design of jobs. It also covers the description of the target audience for the system, e.g. age, education, culture, etc..

Organisation: The inter-relationships between people performing various jobs to further the organisations goals. These may or may not interact directly with the system.

Training: All aspect of the identification of training needs and the provision of training and user support.

Maintenance: All aspects relating to the maintenance of the system.

Environment: The workplace of the users of the system and the workspace within which they undertake their work.

MUSE is directed at the Human Computer Interaction technical area. By examining the interaction between each of the technical areas with the Human Computer Interaction area the scope and potential of MUSE can be explored.

Human Computer Interaction - Equipment. MUSE provides for a good exchange of HCI and domain related information. It does not identify the impact of software architecture or hardware constraints on the design process.

Human Computer Interaction - User. MUSE describes user tasks (both on and off-line) as a result it can directly contribute to the design or modification of user jobs. The design of the Human Computer Interaction should take account of different demands made by a variety of users performing different jobs. MUSE has no concept of task assignment to users to form jobs. It does not address large multi-user systems. Additionally MUSE has no concept of a description of the system users and therefore has no way of enabling this type of information influence the design.

Human Computer Interaction - Organisation. The nature of the organisation, its structure and working practices, is not taken into account within MUSE. There is no consideration of the effects of the introduction of the software system on the organisation.

Human Computer Interaction - Training. In describing the tasks to be performed and providing detailed specification of the HCI MUSE could contribute to the identification of training needs and the development of courses. MUSE does not address the design impact of differing user skill levels. MUSE does not explicitly address issues relating to the provision of on-line help or the application of computer-based training. Both of these introduce specific requirements and place demands on the design of the HCI. There is no reference to user documentation.

Human Computer Interaction - Maintenance. MUSE in isolation could represent maintenance tasks (associated with the HCI for maintainers) and provide for the design of screens. However, in the context of a large system where maintenance is addressed as part of a structured Logistic Support programme much of this task information would be produced. There need to be links between MUSE and other programmes associated with systems development.

Human Computer Interaction - Environment. MUSE takes no account of the environment(i.e. the work place) within which the system is to operate. For example, the need to provide a portable system will place different constraints on the HCI, e.g. to be operated by the keyboard.

Consideration of the interactions above demonstrate MUSE's potential. However, the method would need to be greatly expanded to encompass all of the issues.

4. SUMMARY

It is our conviction that the SPI community has yet to give due recognition to the importance of human factors techniques in the analysis and design stages of software projects. The inclusion of human factors as an integral component of the software process is vital for the design of effective systems. This integration has been hampered by the problems in team communication and design specification problems. We have described our experience in using the Method for Usability Engineering which addresses these problems.

The evaluation of MUSE undertaken in the study extends further than the issues discussed in this paper. MUSE offers a number of benefits but its scope is highly constrained and this leads to some significant shortfalls. These shortfalls impact on the design of the HCI for which MUSE is intended and the wider use of human factors within system design. However, within the context of integrating human factors into software engineering MUSE is one of the few published methods. As such it does offer a number of benefits and is not so constrained as to prevent its development.

The MUSE process introduces a rigour to the application of human factors within the context of system design. The identification of distinct products within the context of the system life cycle begins the process of formally identifying when and how human factors contributes to the design. More importantly it allows this contribution to be tested.

Aiding Communication

The use of the structured notation provides a vehicle through which the communication of human factors design ideas can be conveyed to software engineers. Any mechanism that aids this line of communication is of benefit. The structured notation offers the opportunity for two way communication. The same notation can be used by the software team to convey information on the system architecture.

HCI Specification

The set of products provided by MUSE represent a comprehensive specification for the design of the HCI. It addresses the dynamic feel of the design in addition to its static 'look'.

Wider Contribution of Human Factors to Effective Systems

MUSE's scope is focused exclusively on the design of the HCI. In the context of system development, the results of the human factors activities undertaken to support MUSE would support additional input to the design of an effective system:

- The task descriptions would support the identification of training needs;
- The HCI specification would support the generation of user documentation and training course material;
- The task descriptions would support consideration of job design and organisational change.

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 that will entail revisions to MUSE, or contribute to the development of a new approach.

Acknowledgements

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

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.

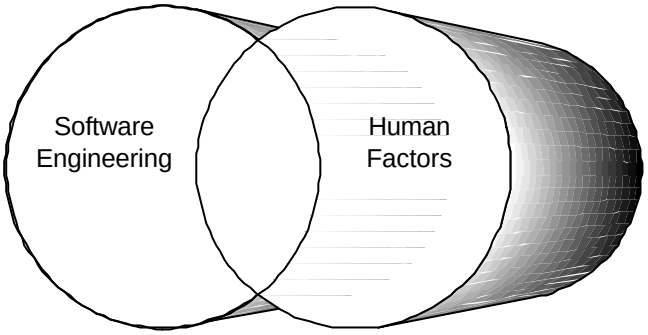

Figure 1: The Intersection of Software Engineering and Human Factors
Defining when and how the two disciplines should interact

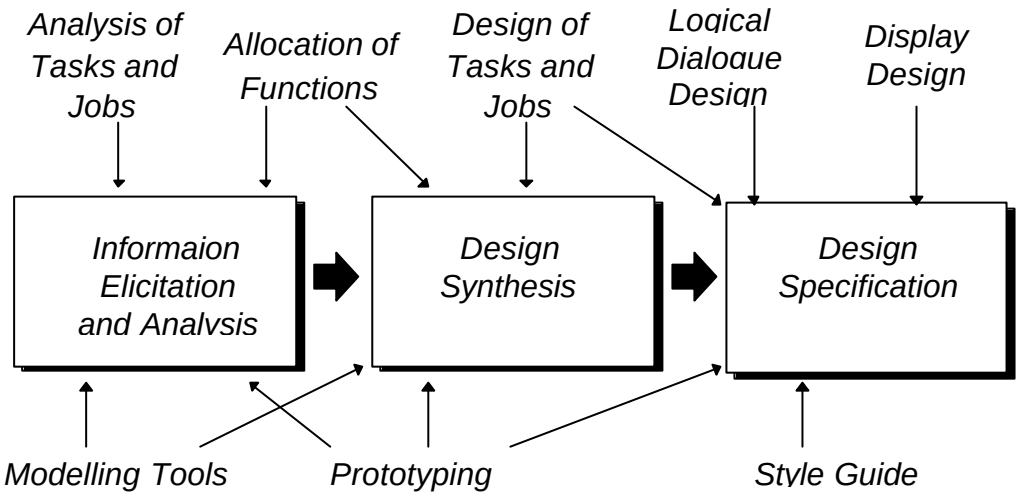

Figure 2: The Three Phases of MUSE
MUSE allows for the importing of existing human factors techniques and tools.

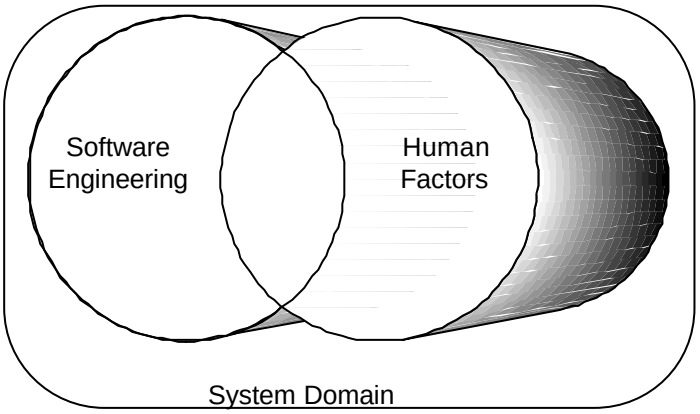

Figure 3: The Intersection of Software Engineering and Human Factors
The development of the system must take account of the domain within which it will operate.

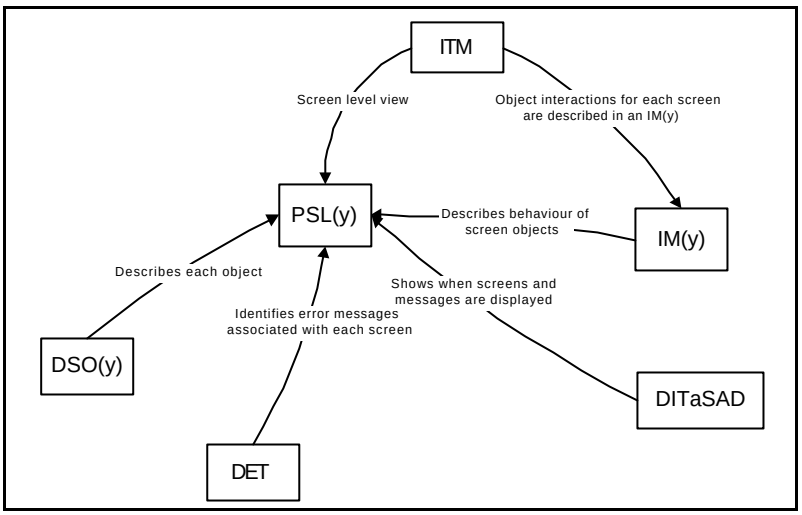

Figure 4: The MUSE Specification Of The System HCI

The task-based approach to representing the dialogue of the system allows the components of the specification to be reviewed by software engineers and user representatives using a walk through approach.

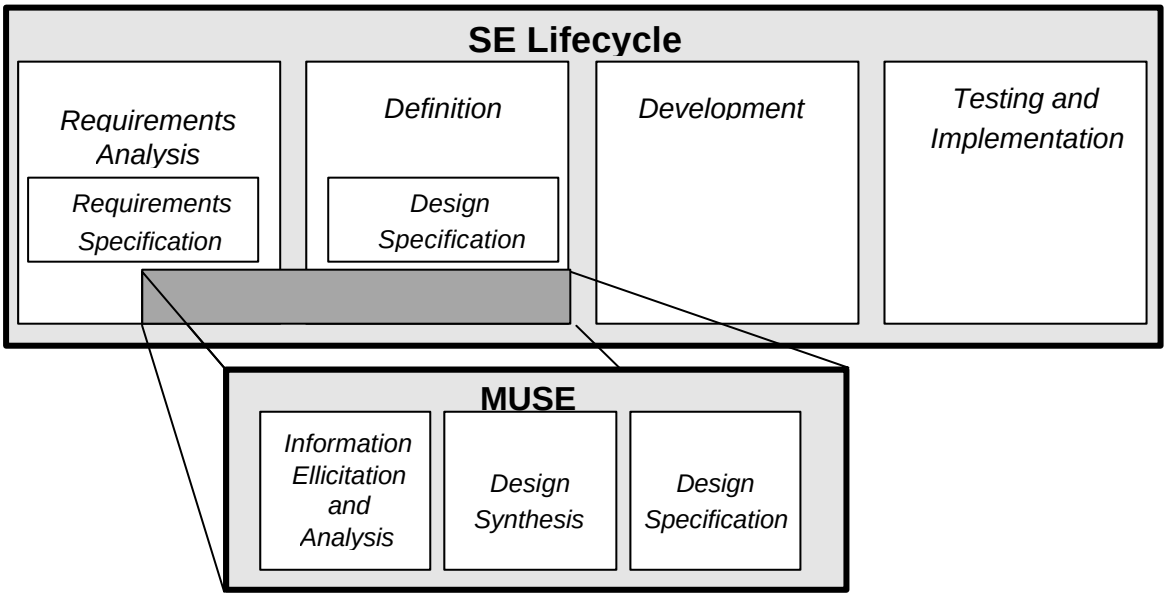

Figure 5: Mapping of MUSE onto Traditional System Life cycle

MUSE currently does not extend throughout the software life cycle to completely address requirements or evaluation of the resultant design.

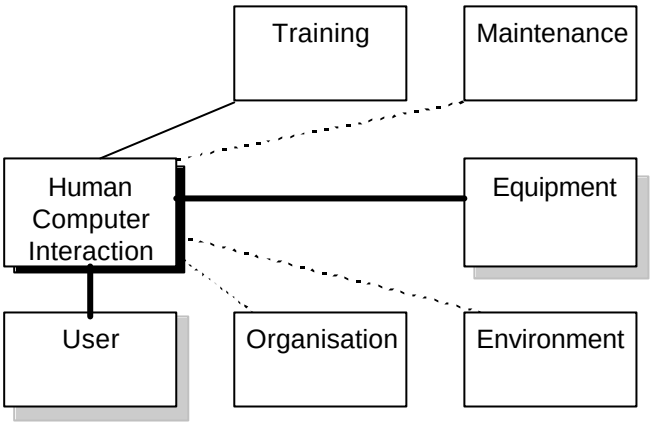

Figure 6: Diagrammatic Representation of the scope of MUSE and its Relationship to other Technical Areas
Boxes with shadows represent where MUSE provides some coverage. Black shadows represent virtually complete coverage; Grey shadows represent only partial coverage.
The bold lines show where MUSE provides a two-way link; The solid lines represent where MUSE could currently contribute but does not receive any input; the dotted line represents where MUSE currently has no links.

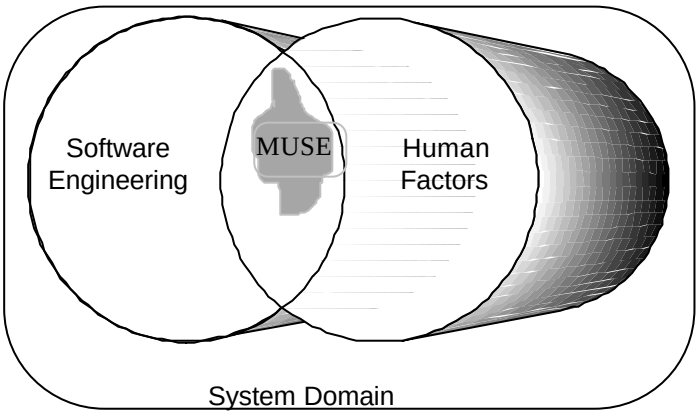

Figure 7: The Intersection of Software Engineering and Human Factors: the Human Computer Interface
MUSE represents only a partial intersection of Software Engineering and Human Factors