Synthesis-Oriented Situational Analysis in User Interface Design

Kevin A. Mayo and H. Rex Hartson

TR 93-20

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

June 11, 1993

Accepted in East-West International Conference on Human-Computer Interaction, Moscow 1993

Synthesis-Oriented Situational Analysis in User Interface Design

Kevin A. Mayo & H. Rex Hartson

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
(703) 231-4857 (703) 231-6931
lastname@cs.vt.edu

Abstract. *Analytic evaluation* is a term describing a class of techniques for examining a representation of a user interface design, and discovering design flaws and/or predicting user task performance. In our work with analytic evaluation, we have observed limitations on the effectiveness and efficiency of analytic techniques for formative evaluation supporting the iterative design and re-design cycle. Here we support those observations with arguments based on theoretical limitations of the models underlying these techniques. By way of comparison we discuss desirable characteristics for an alternative approach. In our search for such an alternative, we have developed the Task Mapping Model, a substantively different approach to analysis for supporting the user interface design. We briefly describe the Task Mapping Model and give some examples illustrating its desirable characteristics.

Keywords: User Interface Design, User Interface Design Requirements, User Interface Evaluation, Task Description, Task Analysis.

1 Introduction

Analytic evaluation refers to a class of techniques based on examining a description of a user interface design, often before it is prototyped or implemented, in order to detect usability problems or to predict user task performance. Many of the predictive models associated with these techniques are validated empirically. Analytic evaluation techniques are usually intended as a substitute for early user testing and/or a supplement for later user testing.

One measure of *effectiveness* of a formative evaluation technique is the number of usability problems that can be found in a given design. A related measure, *efficiency*, can be thought of in terms of the number of usability problems identified per unit of designer effort. A measure based on usability problems identified *and solved* per unit of designer effort might be even more useful, since it helps to close the iterative loop by connecting to the redesign process within formative evaluation. These are the kinds of measures we have in mind as we discuss the role of analysis in formative evaluation.

While recently exploring existing analytic evaluation techniques and developing some of our own, we have concluded that most existing models for analytic evaluation are ineffective and inefficient as methods for formative evaluation in an iterative development cycle, for reasons based on the theoretical limitations of the underlying models. We are developing the Task Mapping Model, a model for analyzing usability problem situations highlighted by the empirical formative evaluation process. Knowledge needs that users have for performing tasks are determined, resulting in clear and specific design/redesign requirements for solutions. This papers outlines why we believe that the Task Mapping Model, when used in conjunction with established empirical formative evaluation methods, is inherently more efficient than many commonly known analytic evaluation techniques.

2 Background and Related Work

As increasingly intricate computer systems are employed in every area of our society, many people find themselves reluctant computer users. To assist these users in performing computer tasks, human-computer interaction researchers and specialists have devised methodologies and models for aiding the design of complex computer interfaces. These models differ in form and function, including:

Analysis of user knowledge (e.g., Action Language [27]; TAG: Task-Action Grammar [26]; TAKD: Task Analysis for Knowledge Descriptions [13]; TKS: Task Knowledge Structures [12]),
Performance prediction (e.g., GOMS: Goals, Operators, Methods, and Selection Model [2]; Keystroke Level Model [2]; Cognitive-Complexity Theory [15]), and
Interface modeling (e.g., CLG: Command Language Grammar [22]; ETAG: Extended Task-Action Grammar [29]; ETIT [23]).

Most of these involve task descriptions, user mental models, and models of user knowledge for task performance in one way or another. Overviews of these models can be found in [30, 7].

Reisner's Action Language Model [27] is based on a grammatical analysis of a formal grammar description of an interface design. Actions and inputs are viewed as expressions in the language of the user. Complexity measures are applied to grammars of the user's language, revealing inconsistencies, predicting user performance, and identifying design decisions that might cause user errors. The Action Language models computer and user as two cooperating and communicating information processors.

The GOMS model [2] provides a foundation based on psychological theories for purposes of predicting user performance. Complex cognitive tasks are encapsulated within the operators to simplify the modeling. The amount of detail generated in a GOMS interface description allows for a very thorough evaluation at a very low level of detail, but the GOMS description can be an enormous, difficult, and tedious to produce, and typically requires the skills of a trained psychologist.

Using the GOMS model as a basis, Kieras and Polson [14, 15], have built a formal model to describe user knowledge required for the performance of a task and for the use of a device. This model uses computer program-like descriptions, written as sets of production rules of user tasks that are analyzed or run as simulations to predict user performance complexity. Also, Kieras in [16] attempts to relieve the need for psychological training in the NGOMSL model.

The Command Language Grammar approach to user modeling [22] hierarchically decomposes system functions into objects, methods, and operations. The psychological hypothesis underlying the CLG is that ". . . to design the user interface is to design the user's model" [21]. Idealized user knowledge is represented with a somewhat complex grammar having the appearance of a high level programming language. As with GOMS, creating a CLG description is complicated and time-consuming.

The Task Action Grammar [26] is another formal user model—specifically, a cognitive competence model—with a command-language orientation. A meta-language of production rules encodes generative grammars that convert simple tasks into action specifications. As in Reisner's work, a goal of TAG is to capture the notion of consistency. Marking of tokens in production rules with semantic features of the task allows representation of family resemblances, a way of capturing generalities of which the user may be aware [30]. Complexity measures taken on the production rules are predictors of learnability.

Our own perspective in this paper is substantively different from the above approaches, but has several points in common with the work of Carroll and Rosson; we especially recommend their book chapter on usability specification [5]. Like our Task Mapping Model and unlike the predictive models for analytic evaluation, Carroll and Rosson's [3, 4] Task Artifact Framework has a significant empirical flavor. Carroll and Rosson contend that interface designs contain artifacts, user interface objects, that are given meaning by user tasks. The artifacts represent various theories through which properties of the artifacts imply certain psychological consequences with regard to how users will view and use them. The interface designs incorporate certain claims, which are articulated with both positive and negative (Carroll and Rosson's upside and downside) clauses, about how the artifacts support the associated tasks. Later, during evaluation, situated empirical evidence will arise tending to confirm or refute the theory, and redesign, in the context of psychological design rationale, may be necessary as part of the iterative process. The Task Mapping Model also provides a framework in which situations (e.g., critical incidents identifying usability problems associated with tasks) resulting from formative evaluation are analyzed to produce interface design/redesign requirements for supporting user needs.

There is also a class of formative evaluation techniques not based on either formal analysis or empirical user testing. These *review-based* evaluation procedures include cognitive walk-throughs [18], other kinds of walk-throughs [1], heuristic evaluation [24], and other *usability inspection* methods [19]. We believe review-based methods have potential as efficient techniques for formative evaluation; however, the topic is outside the scope of the present paper.

3 Theoretical Limitations of Analytic Evaluation Techniques

The limitations in the ability of current analytic evaluation techniques to support the iterative design cycle point directly to desirable characteristics for a more effective approach. The analytic evaluation techniques we are discussing include techniques associated with GOMS, CLG, TAG, TAKD, Kieras and Polson's approach, and Reisner's Action Language Model. In grouping these techniques together, we recognize that they are different techniques and do not all have the same characteristics. However, space does not permit making comparisons one by one, with specific references to the literature. Thus, statements made about limitations apply to the group in general, but each statement does not necessarily apply to every technique.

3.1 Analysis vs. Synthesis Orientation

- Current: Most of the analytic evaluation techniques were originally oriented toward analysis, intended to analyze existing interface designs. These methods do not directly support iterative design. To improve an existing interaction design, a developer must create a new design idea, modify the design model, and see if the analysis shows the design to be better.
- Desired: We seek a synthesis orientation, that could be used to capture new interface designs as they occur and to aid the process of creating new designs.

3.2 Performance Prediction vs. Design Support

- Current: Many of the analytic evaluation techniques have as a primary goal user performance prediction, and successes have often occurred in special cases where saving a keystroke or two can make a difference. Performance prediction can be used only to compare two designs and, in itself, can suggest nothing to improve a user interface design.
- Desired: We have a goal of more direct design support. In particular, we seek a technique that can produce design/redesign requirements.

3.3 Essentially Analytical vs. Essentially Empirical

- Current: The analytic evaluation techniques are based on manipulating design representations and cannot take advantage of empirical data from users.
- Desired: We seek an approach that involves analysis, but is still essentially empirical, able to draw on the strength of empiricism inherent in the iterative design cycle. This distinction is made cogently by Carroll and Rosson [5], as they point out that iterative design itself is essentially empirical.

3.4 Error-free Expert Performance vs. Error Handling

- Current: Almost all of the analytic evaluation techniques are based on the assumption of error-free, expert task performance.
- Desired: We seek an approach that considers every task to be a potential error site. As Carroll and Rosson [5] have said, error handling is an essential part of task performance. In fact, error handling itself involves real tasks for interpreting and reacting to feedback, which must also be accounted for in the interface design. The study of error-related situations is essential in designing for usability.

3.5 Task Hierarchies Only vs. Temporal Relations

- Current: While most of the analytic approaches produce a hierarchically decomposed task structure, none deal with the temporal relations necessary to provide a complete task-oriented representation of a design.
- Desired: We seek an analytic model in which task descriptions are built upon temporal relations. We need to be able to represent task performance in contiguous time, task interruption, interleaving of tasks in time, and the inter-relationships of concurrent tasks.

3.6 Global Modeling vs. Situational Analysis

- Current: The analytic evaluation techniques are global modeling techniques, requiring enormous effort and detail in modeling the entire design before analysis, at any level, can be done.
- Desired: We seek a more *situational* approach that, like the analysis in Carroll and Rosson's task artifact framework, can be applied where it is needed the most—at specific situations or trouble spots identified by formative evaluation—without first having to model the entire system in detail.

3.7 Closed-loop vs. Open-loop Interaction

- Current: Most of the analytic evaluation techniques are oriented toward open-loop interaction, where a command is issued and the system executes it.
- Desired: We seek an approach oriented toward closed-loop interaction, which includes direct manipulation and incremental user planning within a tight loop of concurrent action, feedback perception, and adjustment.

3.8 Consideration of Different User classes

- Current: Many analytic evaluation techniques cannot take different user class definitions into account.
- Desired: We seek a model for analysis in which user class definitions constitute an explicit separate dimension. The same task situation will yield different design/redesign requirements for different user classes.

3.9 Dependency on User Mental Models

- Current: Most of the analytic evaluation techniques are built upon mental models of users. While this facilitates the application of cognitive psychology to the modeling, so far the mental models used have not been powerful enough to provide direct solutions to many of the usability problems encountered in the interface development process. The result is a dependency on mental models, without having sufficient descriptive power of mental processes.
- Desired: We seek an analytic approach that does not depend on user mental models, but that can instead make creative use of empiricism to decide how users view tasks.

4 The Task Mapping Model (TMM)

While the TMM can be used for initial design, our discussion here is limited to using the TMM during analysis and re-design of user interface designs.

4.1 Supporting the Iterative Interface Design/Re-design Process

Currently we perceive a gap between the formative evaluation phase and re-design phase of user interface design. There is a lack of methodological support for translating formative evaluation findings into new design requirements or solutions. TMM provides methodological support for bridging this gap, as seen in Figure 1. In a nutshell, TMM allows specialists to analyze users' tasks and system interfaces to synthesize new interface design requirements. It is worthwhile to note that TMM analyses produce new interface design requirements and *not* interface design solutions. It is not our intention to tie the hands of the designers, only to aid them in focusing their creativity to meet specific requirements.

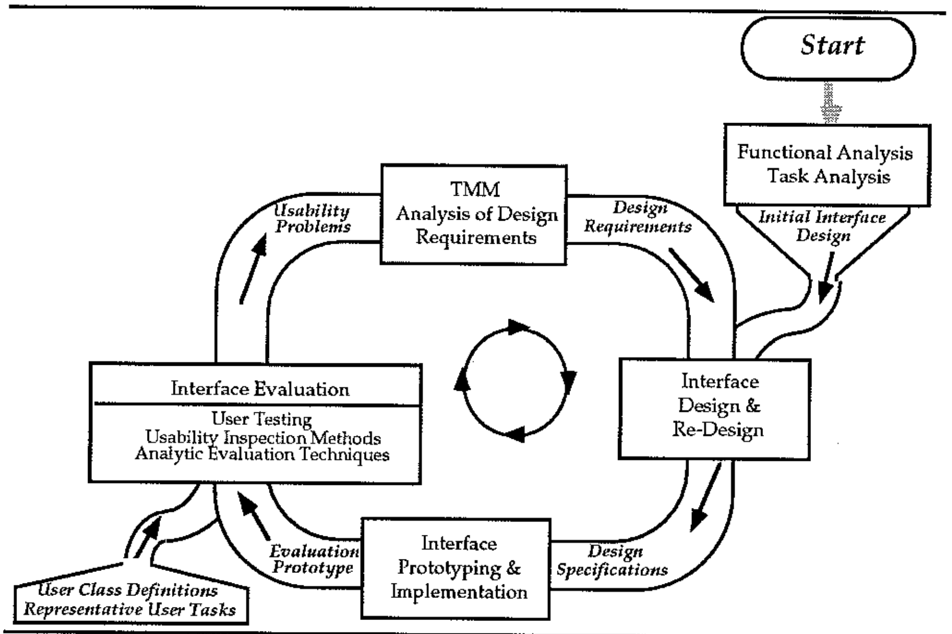


Figure 1. Simplified Representation of Iteration within the Interface Design Process

4.2 Task Performance with Computers

It is a basic premise within the Task Mapping Model that the use of computers to perform tasks requires each task to be conceptualized in various domains of abstraction. These domains are viewed as levels (task, semantic, syntactic, and interaction) in CLG [22], which is a model of interaction as well as a model of the interface. Shneiderman, in his model of long term user knowledge [28] calls them the task semantic, computer semantic, and syntactic domains. Since all of the domains contain tasks (just at different levels of abstraction), we prefer to call this first domain the *Problem Domain*. This is the domain in which the user deals with the problem to be solved independently of computer-related considerations.

Figure 2 shows the domains of abstraction that comprise the TMM framework for describing task performance. Each domain contains domain items (actions, objects, and sub-tasks represented in the figure as open circles) associated with its level of abstraction.

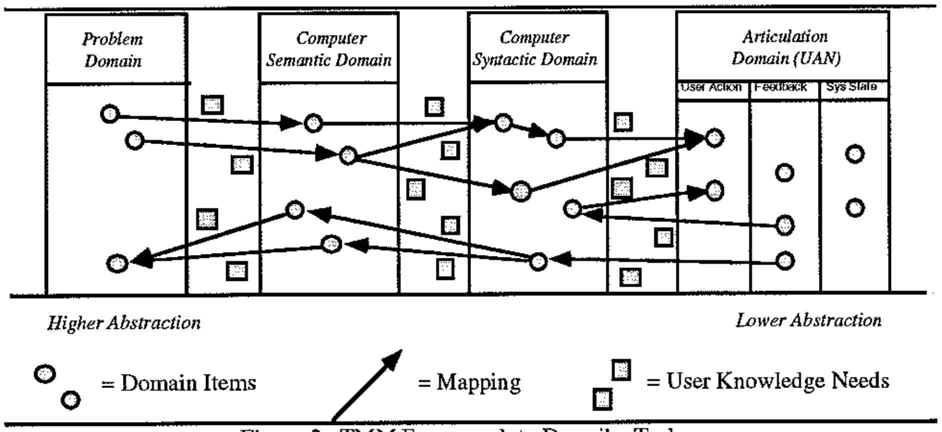


Figure 2. TMM Framework to Describe Tasks

We make three observations about this framework as a model of task performance. Our first observation is that *the user must map* each task to be performed using a computer from one domain to the next, starting from the problem domain, which is known to the user, and going to the computer articulation domain, which contains the necessary physical interactions. Each mappings is shown as arrow in Figure 2, transforming an individual domain item from one domain to the next. The net effect of the mappings between domains is a reconceptualization, by the user, of the task from one domain to the other. The small squares associated with each mapping represent user mapping needs, i.e., knowledge users need to make the mappings. Mappings from higher to lower levels of abstraction correspond to Norman's *execution paths*. [25, 11]

The second observation is that the mappings also exist in the opposite direction—also shown in Figure 2. Originating with the computer feedback in the articulation domain, these mappings represent the abstraction of behavior occurring within interface artifacts back to the problem domain. These mappings from lower to higher levels of abstractions are equivalent to Norman's *evaluations paths*. [25, 11]

The third observation the articulation domain is the site of the User Action Notation (UAN), a task-oriented notation we have developed for behavioral representation of user interface designs [8, 9]. The UAN is based on user actions, system feedback, and interface state changes that occur in the articulation domain.

4.3 TMM Task Descriptions

As discussed above, TMM task descriptions are based on a framework within which one can decompose tasks into levels of interaction or abstract domains. This decomposition framework aids well the identification of user knowledge necessary for making the mappings and performing the task, and also provides an indication of

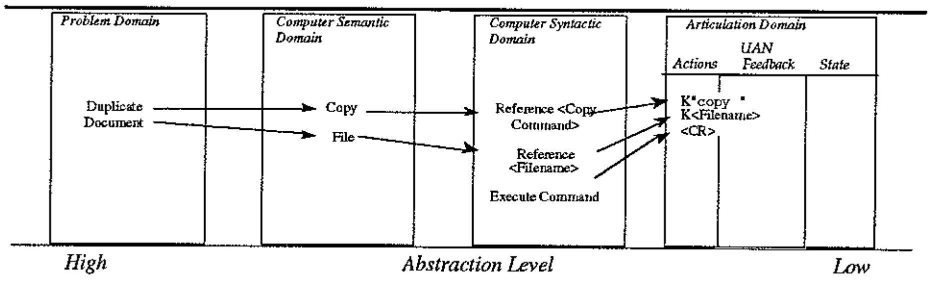


Figure 3: Example of TMM Domains with Mappings

the level of abstraction corresponding to a given element of this user knowledge. To elaborate on the four domains that comprise TMM's task description framework:

Problem Domain: Highest level of abstraction that includes items defined within the users' work world, external to the computer system,
Computer Semantic Domain: Highest level of abstraction of computer items involved in a given task being modeled,
Computer Syntactic Domain: Grammatical components of interaction, including commands, computer objects, and associated user actions. Although more abstract that the specific physical user actions of the articulation domain, these syntactic components are somewhat dependent on interaction style (e.g., action-object order *verses* object-action order, or 'select command' for direction manipulation *verses* 'reference command' for a command line interface).
Articulation Domain: The actual communication sequence between user and computer. (TMM currently employs the user action notation, UAN [9], as a behavioral notation to describe user action and feedback at the articulation level.)

Figure 3 depicts a command line interface example task (`DUPLICATE DOCUMENT`) mapped from the problem domain down to the articulation domain. It should be noted the computer syntactic domain terminology is derived from Lenorovitz's et al. *user-internal* and *user-external* taxonomies. [17]

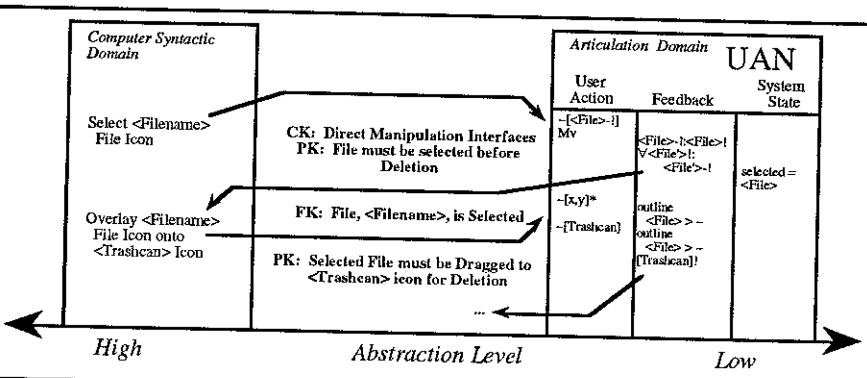
In order to make the mappings, users have knowledge requirements. For example, to perform the mapping between `DOCUMENT` and `FILE`, the user must know that a `DOCUMENT` is stored in an abstract computer artifact called a `FILE`. Each mapping is potentially associated with a set of knowledge elements—indicating the users' knowledge needs for that mapping. TMM knowledge elements are not specified within a formal grammar, but in natural language.

The TMM distinguishes different types of knowledge into the following categories:

Factual Knowledge (FK): Specific facts. E.g., documents are contained in files, files can be grouped into folders.
Conceptual Knowledge (CK): Collections of knowledge that comprises a whole. E.g., general knowledge about direct manipulation interfaces, pointing devices.

Procedural Knowledge (PK): Course of action and outcome knowledge, i.e., knowledge that indicates how to do something necessary for task performance. E.g., double-clicking on a folder shows contents.

The example in figure 4 shows part of a TMM description for the task DISCARD DOCUMENT (problem domain) within the Macintosh environment (computer syntactic domain). Notice that the system feedback starts an evaluation path from the articulation back to the computer syntactic domains, and thus, interaction specialists can incorporate feedback within their analysis. This example also shows various knowledge elements associated with the paths. Of course, this example is only a small piece of the overall description for DISCARD DOCUMENT.



Computer Syntactic Domain
Select <Filename> File Icon
Overlay <Filename> File Icon onto <Trashcan> Icon
CK: Direct Manipulation Interfaces
PK: File must be selected before Deletion
FK: File, <Filename>, is Selected
PK: Selected File must be Dragged to <Trashcan> icon for Deletion
...
Articulation Domain
UAN
User Action
Feedback
System State
~[<File>-!] Mv
<File>-!:<File>! ∀<File'>!: <File'>-!
selected = <File>
~[x,y]*
~[Trashcan]
outline <File> > ~
outline <File> > ~ [Trashcan]!
High
Abstraction Level
Low

Figure 4: Example Task Description with Knowledge Needs on Mappings

Creating a full TMM task description for a particular task is an iterative process. First a task is chosen, and decomposed as far as possible within the problem domain. (It is possible that only one of its sub-tasks needs to be analyzed.) Next, the task is represented within the computer semantic and syntactic domains, and finally the articulation domain as the user would perceive it. These domains are not foreign to the task analyst, but the notation and mappings among the domains are new. The analyst links related adjacent domain items with mappings and identifies necessary user knowledge for each mapping. This process continues until the analyst is satisfied with the description; satisfaction is attained through task description walkthroughs and peer evaluation.

The *TMM Analysis Guide* [20] further describes notations for knowledge types, variables within task descriptions, and task timing relationships (e.g., sequential, interleaved, concurrent).

4.4 TMM Life Cycle

The discussion, thus far, has centered on TMM as a task description technique; however, TMM consists of several methods for describing and analyzing tasks to derive interface re-design requirements. The process begins by focusing on a single situation (thus, we call it *situational analysis*) involving a usability problem identified during formative evaluation. Within the TMM framework, the analysts describe just those user tasks directly related to the usability problem in question, decomposing

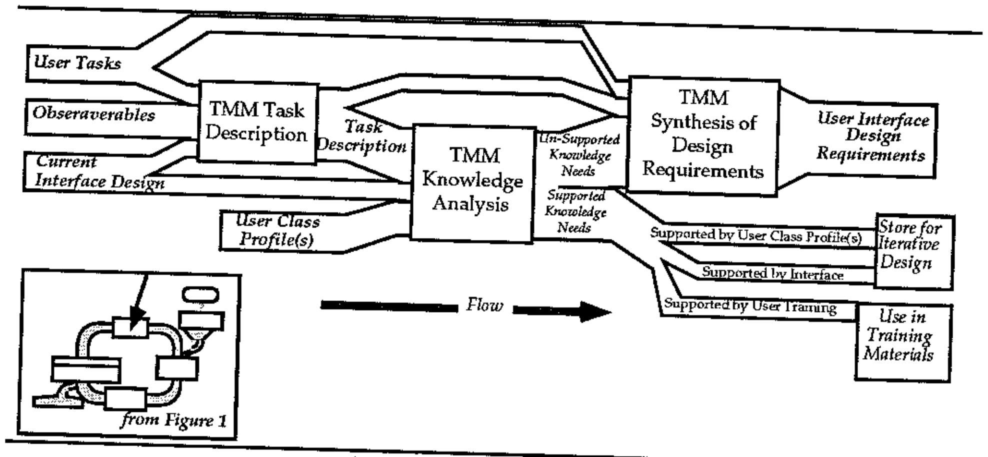

User Tasks
Obseraverables
Current Interface Design
User Class Profile(s)
TMM Task Description
Task Description
TMM Knowledge Analysis
Un-Supported Knowledge Needs
Supported Knowledge Needs
TMM Synthesis of Design Requirements
User Interface Design Requirements
Supported by User Class Profile(s)
Supported by Interface
Supported by User Training
Store for Iterative Design
Use in Training Materials
Flow
from Figure 1

Figure 5. TMM Life Cycle

them over the domains of the model. By analyzing the task description, the designer can determine the total knowledge required by the user to map the task through the domains.

User class profile(s) established for a design, which describe the skills and knowledge users are expected to bring to a task, are used to determine which knowledge requirements can be supported by the user. These are subtracted from the total knowledge needs for the task. Similarly, knowledge already supported in the interface design and knowledge expected to be supported by training can also be subtracted from the total, as shown in the right-hand side of figure 5. The remaining knowledge requirements are unsupported and our experience has been that, by now, these needs are specific enough to be stated almost directly as re-design requirements. Within the broader interface development life cycle these requirement are given to designers who will produce designs, which are passed on to the implementer, and so on back through the iterative cycle of design and evaluation.

TMM provides analyses to derive only interface design requirements and not interface design specifications. This is an important distinction because TMM will not stunt the creativity of the interface designers, only point them to unsupported knowledge needs. This process of task description, analysis for user knowledge needs, and synthesis of interface design requirements represents the TMM life-cycle—depicted in Figure 5.

A complete description of the TMM life cycle, and its component parts, is in [20].

4.5 Example

Limited space permits only a brief, simple example to illustrate the TMM life cycle and the production of re-design requirements. Space limitations also preclude the use of forms which have been developed to aid TMM analysts in carrying out this process. Here we can only summarize.

Consider, as part of a broader word processing task, that a user wishes to discard (remove from the system) a specific existing document. In the problem domain, this is an instance of the DISCARD DOCUMENT task that we discussed in section 4.3. In order to make this mapping the user must know that DOCUMENTS on the computer are stored in FILES(factual knowledge, perhaps as part of a body of conceptual knowledge). The user must also know that there is some kind of command (that we are generically calling ERASE here) to remove a FILE within the computer system. So the mapping for this task from problem to computer semantic domain is a mapping of DISCARD DOCUMENT to ERASE FILE. Suppose the user class definition makes it clear that the users we are focusing on should know that documents are stored in files and that files are discarded by erasing, then no re-design requirements are generated.

Next we look at the mapping from the computer semantic to the computer syntactic domain. Since, in the syntactic domain, we are no longer independent of the interaction style of the implementation platform, let us assume we are using an Apple Macintosh computer. The syntax involves a sequence (as seen in Figure 4): SELECT <FILENAME> FILE ICON, followed by OVERLAY <FILENAME> FILE ICON ONTO <TRASHCAN> ICON. A user familiar with the Macintosh will have no problem mapping this to the articulation domain as shown in Figure 4. However, if the user is only a little familiar with desktop-metaphor computers, the knowledge of how to map the ERASE command to the dragging and dropping of the file icon into the trashcan could be missing. In such a case the Macintosh design does not support the users' knowledge needs for this task.

Looking in the user class profile, suppose we determine that, although the user cannot be expected to know the exact command name for erasing, the user will have knowledge the concept of erase (conceptual knowledge). If, however, the profile identifies a general understanding of a WIMP (window, icon, menu, pointer) interface, it is reasonable to assume users in this situation will know how to search for a command name that matches the concept of erase (e.g., by looking at all button labels and through all menus) and, with reasonable command names, will recognize the match when it is found. If the system being used is, say, Microsoft Windows, our analysis leads us to conclude that the user will discover the DELETE command on the FILE menu and the task can be completed. Using the Macintosh, which has no explicit DELETE command for files, the user will not be supported.

This leads directly to the following re-design requirement for this user class and this task: provide an explicit visual cue to aid discovery of the file deletion command. The way in which this requirement is satisfied is the responsibility of the designers and outside the realm of TMM, but it could be something as simple as including a DELETE choice on the FILE menu. To aid learning by the novice user of the drag and drop alternative, the design could also provide animation of the file icon moving over and into the trashcan icon as part of the feedback for the DELETE choice from the FILE menu.

5 Conclusions

In our search for an alternative analysis technique for supporting the iterative interface design process [6], we have developed the Task Mapping Model, a substantively different approach. In comparison to the theoretical limitations of most of the established approaches to analytic evaluation, we find the Task Mapping Model more suitable for combining analysis with empirical formative evaluation, as a method for supporting iterative user interface design. Our early experience verifies that this approach possesses most of the desirable characteristics we set as goals for an alternative approach:

- The Task Mapping Model has a synthesis orientation, intended to capture new interface designs as they occur and to aid the process of creating new designs. The Task Mapping Model shares the task orientation of the analytic evaluation models, but is more synthesis-oriented, because the underlying design representation technique (the User Action Notation) was created specifically to communicate interface designs to implementers.
- The Task Mapping Model has a goal of direct design support, rather than user performance prediction, yielding specific design/redesign requirements.
- The Task Mapping Model is essentially empirical in that it is explicitly designed to take advantage of the empirical data generated by formative evaluation and can use empirical observations as models for how users perform tasks.
- Not being limited to error-free, expert task performance, the Task Mapping Model considers every task as potentially an error site. Error handling is described and analyzed just as any other task.
- The Task Mapping Model task descriptions are more complete because they include the temporal relations of the UAN.
- Although the Task Mapping Model can be applied globally to an interface design, a strength of this approach is in situational analysis that can be applied where it is needed the most. Trouble spots identified by formative evaluation can be analyzed without first modeling the entire design.
- The Task Mapping Model is oriented toward closed-loop interaction, which includes direct manipulation and incremental user planning.
- Different user class definitions are an explicit dimension of the Task Mapping Model which uses these definitions to determine very specific user needs.
- The Task Mapping Model does not depend on user mental models (i.e., models of how users store and retrieve information).

TMM fills the gap between formative evaluation and interface design. TMM is a practical task-oriented approach that utilizes situational analysis to derive new user interface design requirements. As a methodology, TMM is still in its infancy and needs empirical validation which is planned as future work.

There are many other avenues of future research for TMM. Task metrics are an obvious extension because of TMM's rich task descriptions. Training is also an area where TMM could be applied. The generated lists of necessary knowledge can provide the bedrock information of user training programs. Moreover, because of TMM's readability, its task descriptions can be used for documentation.

Acknowledgments

The authors wish to acknowledge the efforts of: Helen Crawford, Sallie Henry, Lucy Nowell, Bill Wake, Robert and Beverly Williges, and the members of the Virginia Tech HCI research group. Ideas expressed in section 3 are also found in [10]. Partial funding of this research was provided by the National Science Foundation, with guidance from John Hestenes. We also wish to extend a special *thank you* to Debby Hix for her insightful comments and continued help with this research.

CR Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Tools and Techniques—*user interfaces*; D.2.1 [**Software Engineering**]: Requirements and Specifications—*methodologies*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*evaluation/methodology, screen design, theory and methods*

References

1. Bias, R., *Walkthroughs: Efficient Collaborative Testing.* IEEE Software, 1991. **8**(5): pp. 94-95.

2. Card, S. K., T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction.* 1983, Hillsdale, New Jersey: Lawrence Erlbaum Associates.

3. Carroll, J. M. *Infinite Detail and Emulation in an Ontologically Minimized HCI,* in *Human Factors in Computing Systems, CHI '90 Conference.* 1990. Seattle, Washington, April 1-5: ACM, pp. 321-327.

4. Carroll, J. M., W. A. Kellogg, and M. B. Rosson, *The Task-Artifact Cycle,* in *Designing Interaction: Psychology at the Human-Computer Interface,* ed. J.M. Carroll. 1991, Cambridge University Press: New York. pp. 74-102.

5. Carroll, J. M. and M. B. Rosson, *Usability Specifications as a Tool in Iterative Development,* in *Advances in Human-Computer Interaction,* ed. H.R. Hartson. 1985, Ablex Publishing Corporation: Norwood, New Jersey. pp. 1-28.

6. Gould, J. D. and C. Lewis. *Designing for Usability—Key Principles and What Designers Think,* in *Human Factors in Computing Systems, CHI '83 Conference.* 1983. Boston, Mass., December 12-15: ACM, pp. 50-53.

7. Haan, G. d., G. C. v. d. Veer, and J. C. v. Vliet, *Formal Modelling Techniques in Human-Computer Interaction.* Acta Psychologica, 1991. **78**: pp. 27-67.

8. Hartson, H. R. and P. D. Gray, *Temporal Aspects of Tasks in the User Action Notation.* Human-Computer Interaction, 1992. **7**: pp. 1-45.

9. Hartson, H. R., A. C. Siochi, and D. Hix, *The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs.* ACM Trans. on Info. Sys., 1990. **8**(3): pp. 181-203.

10. Hix, D. and H. R. Hartson, *Formative Evaluation: Ensuring Usability in User Interfaces,* in *Trends in Computing: Human-Computer Interaction,* ed. L. Bass and P. Dewan. 1993, John Wiley and Sons: New York.

11. Hutchins, E. L., J. D. Hollan, and D. A. Norman, Direct Manipulation Interfaces, in User Centered System Design, ed. D.A. Norman and S.W. Draper. 1986, Lawrence Erlbaum Associates: Hillsdale, New Jersey. pp. 87-124, Chap. 5.

12. Johnson, H. and P. Johnson, *Task Knowledge Structures: Psychological Basis and Integration into System Design.* Acta Psychologica, 1991. **78**: pp. 3-26.

13. Johnson, P., D. Diaper, and J. B. Long. *Tasks, Skills and Knowledge: Task Analysis for Knowledge Based Descriptions,* in *IFIP Conference on Human-Computer Interaction—Interact '84.* 1984. London, U.K., September 4-7: Elsevier Science Publishers B.V. (North-Holland), pp. 499-503.

14. Kieras, D. and P. G. Polson. *A Generalized Transition Network Representation for Interactive Systems,* in *Human Factors in Computing Systems, CHI '93 Conference.* 1983. Boston, Mass., December 12-15: ACM, pp. 103-106.

15. Kieras, D. and P. G. Polson, *An Approach to the Formal Analysis of User Complexity.* Int. J. Man-Machine Studies, 1985. **22**: pp. 365-394.

16. Kieras, D. E., *Towards a Practical GOMS Model Methodology for User Interface Design,* in *Handbook of Human-Computer Interaction,* ed. M. Helander. 1988, North-Holland: Amsterdam. pp. 135-157, Chap. 7.

17. Lenorovitz, D. R., M. D. Phillips, R. S. Ardrey, and G. V. Kloster, *A Taxonomic Approach to Characterizing Human-Computer Interfaces,* in *Human—Computer Interaction,* ed. G. Salvendy. 1984, Elsevier Science Publishers B. V.: Amsterdam. pp. 111-116.

18. Lewis, C., P. Polson, C. Wharton, and J. Rieman. *Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces,* in *Human Factors in Computing Systems, CHI '90 Conference.* 1990. Seattle, Washington, April 1-5: ACM, pp. 235-242.

19. Mack, R. and J. Nielsen. *(Workshop) Usability Inspection Methods,* in *Human Factors in Computing Systems, CHI '92 Conference.* 1992. Monterey, California, May 3-7: ACM, pp. 691.

20. Mayo, K. A. *Task Mapping Model Analysis Manual* (TR-93-07). 1993. Department of Computer Science, Virginia Tech (VPI&SU), Blacksburg, Virginia.

21. Moran, T. P., *A Framework for Studying Human-Computer Interaction,* in *Methodology of Interaction,* ed. e.a. Guedj. 1980, North-Holland Publishing Co.: pp. 293-301.

22. Moran, T. P., *The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems.* Int. J. Man-Machine Studies, 1981. **15**: pp. 3-50.

23. Moran, T. P. *Getting into a System: External-Internal Task Mapping Analysis,* in *Human Factors in Computing Systems, CHI '83 Conference.* 1983. Boston, Mass., December 12-15: ACM, pp. 45-49.

24. Nielsen, J. *Finding Usability Problems Through Heuristic Evaluation,* in *Human Factors in Computing Systems, CHI '92 Conference.* 1992. Monterey, California: ACM, pp. 373-380.

25. Norman, D. A., *Cognitive Engineering,* in *User Centered System Design,* ed. D.A. Norman and S.W. Draper. 1986, Lawrence Erlbaum Associates: Hillsdale, New Jersey. pp. 31-65, Chap. 3.

26. Payne, S. J. and T. R. G. Green, *Task-Action Grammar: The Model and its Developments,* in *Task Analysis for Human-Computer Interaction,* ed. D. Diaper. 1989, Ellis Horwood Limited: Chichester. pp. 75-107.

27. Reisner, P., *Formal Grammar as a Tool for Analyzing Ease of Use: Some Fundamental Concepts,* in *Human Factors in Computer Systems,* ed. J.C. Thomas and M.L. Schneider. 1984, Ablex Publishing: Norwood, New Jersey. pp. 53-78.

28. Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 1987, Reading, Massachusetts: Addison-Wesley Publishing Company.

29. Tauber, M. J. *ETAG: Extended Task Action Grammar—A Language for the Description of the User's Task Language,* in *IFIP Conference on Human-Computer Interaction—INTERACT'90.* 1990. Cambridge, U.K., August 27-31: Elsevier Science Publishers B.V. (North-Holland), pp. 163-168.

30. Wilson, M. D., P. J. Barnard, T. R. G. Green, and A. Maclean, *Knowledge-Based Task Analysis for Human-Computer Systems,* in *Working with Computers: Theory versus Outcome,* ed. G.C.v.d. Veer, *et al.* 1988, Academic Press: London. pp. 47-87.