A Procedure for Evaluating Human-Computer Interface Development Tools

By Deborah Hix

TR 90-25

FROM Proceedings of the Second UIST Conference,
November 1989, Williamsburg, VA.

A PROCEDURE FOR EVALUATING HUMAN-COMPUTER INTERFACE DEVELOPMENT TOOLS

Deborah Hix
Department of Computer Science
Virginia Polytechnic Institute & State University
Blacksburg, VA 24061
703/231-6199
email: hix@vtodie.cs.vt.edu

1. INTRODUCTION

1.1. Motivation

The computer industry has seen an explosive emergence of human-computer interface development tools -- sometimes called user interface management systems or UIMS -- in the last few years. Commercial software packages and research systems even tangentially related to the area of human-computer interaction now claim to be UIMS. Human-computer interface development tools are themselves interactive systems that support production and execution of the human-computer interface. With their recent proliferation, evaluations and comparisons are constantly done, but without a formal, structured approach. State-of-the-art in evaluation of these tools is based mostly on warm fuzzy feelings and the ever-popular opinion. Addressing these problems is difficult, largely because of the relative newness of such tools, because of the many differents kinds of systems that are called UIMS, and because of their inherent complexity. These tools are complex because human-computer interfaces, which these tools produce, are complex.

An evaluation procedure that uses a standardized technique to produce quantifiable criteria for evaluating and comparing human-computer interface development tools is described in this paper. An empirical validation study, to determine the consistency of ratings produced by this procedure, is also presented. These ratings could be used, for example, as important data for the task of choosing a tool for a particular human-computer interface development environment. The research presented in this paper is one of the first attempts to produce a structured, quantitative approach to evaluating such tools. Our procedure is being used in several commercial interface development environments (e.g., GTE Data Systems [Arble 1988] and McDonnell Douglas [Totten 1989]).

1.2. Related Research

Precedence has been established for using the type of approach on which our tool evaluation procedure is based. Roberts and Moran [1983], for example, produced a methodology for standardized evaluation of text editors. The basis of their approach was classification of potential editing tasks and evaluation along several dimensions, including time to perform tasks, error costs, learning time, and functionality. A replication study of their work [Borenstein 1985] produced recommendations for modifications, including limitations on using a stopwatch, changes in testing expertise, and extension of the functionality dimension. Cohill, Gilfoil, and Pilitsis [1988] developed a methodology used at AT&T for evaluating software packages, particularly commercial systems. Criteria such as performance, documentation, and support were evaluated. Research such as this provided ideas used in the development of our tool evaluation procedure.

2. OVERVIEW OF THE TOOL EVALUATION PROCEDURE

2.1. Description of the Evaluation Procedure

The tool evaluation procedure revolves around "hands-on" use of each tool to be evaluated. After learning the tool, an evaluator completes a detailed 28 page "checklist" form that is organized around two dimensions:

- Functionality of the tool, and
- Usability of the tool.

Functionality is a measure of what the tool can do; that is, what interface styles, techniques, and features it can be used to produce in a target application interface. *Usability* is a measure of how well the tool performs its possible functions, in terms of ease of use (a subjective, but quantitative, rating of how how easy or difficult the tool is to use) and human performance (an objective rating of how efficiently the tool can be used to perform tasks).

The functionality dimension of a tool's evaluation is organized into three sections:

- *Types of interfaces* a tool can support, including
  --*interaction styles* (e.g., menus, forms, typed string inputs, windows)
  -- *features of interfaces* (e.g., animation, types of navigation, defaults, graphics)
  -- *hardware devices* (e.g., list of various input and output devices);
- *Types of support* provided for the general process of interface development (e.g., rapid prototyping, evaluation, libraries, documentation, on-line help); and
- *General characteristics* of a tool (e.g., consistency, integration).

Each of these is further decomposed into items as appropriate (e.g., 14 types of menus, 27 features of interfaces, 24 hardware devices, 9 rapid prototyping items, 7 evaluation items, 4 general characteristics, and so on).

The usability dimension of a tool's evaluation is measured with two methods:

- *Subjective evaluation* measures ease of use for each of the three functionality sections; the evaluator indicates on the form, *only* for each function the tool can produce, whether it was difficult (indicated by a frowning face), adequate (bland face), or easy (smiling face) to use to produce that function. We attempted to reduce individual interpretation by precisely defining each "face."
- *Objective evaluation* measures human performance using the tool to complete suggested benchmark interface tasks that can be customized to a specific working environment (this aspect is still under development and will not be addressed here).

For those types of interfaces that the tool can produce, the evaluator indicates the primary type of specification/implementation technique the tool uses to produce individual functions. Techniques on the form from which the evaluator can choose include:

- *Textual language coding* (e.g., conventional programming language or specialized dialogue language);
- *Direct manipulation* (e.g., objects or graphical "programming"); and
- *Other techniques* (e.g., tables, rules, form-filling).

Figure 1 shows a representative page from the "types of interfaces" section, the page for evaluating "forms".

When the detailed portion of the evaluation form is completed, the evaluator calculates overall functionality and usability ratings (using an electronic spreadsheet that exactly replicates layout of the form) to produce numeric results for an executive summary. Finally, if desired, the evaluator performs benchmark interface development tasks, completing the evaluation.

To summarize, steps in the tool evaluation procedure include:

- Acquisition of software, hardware, and all source materials to be used in the evaluation
- Learning to use the tool(s) to be evaluated
- Completion of the evaluation form
- Performance of benchmark interface development tasks (optional step)
- Computation of functionality and usability scores.

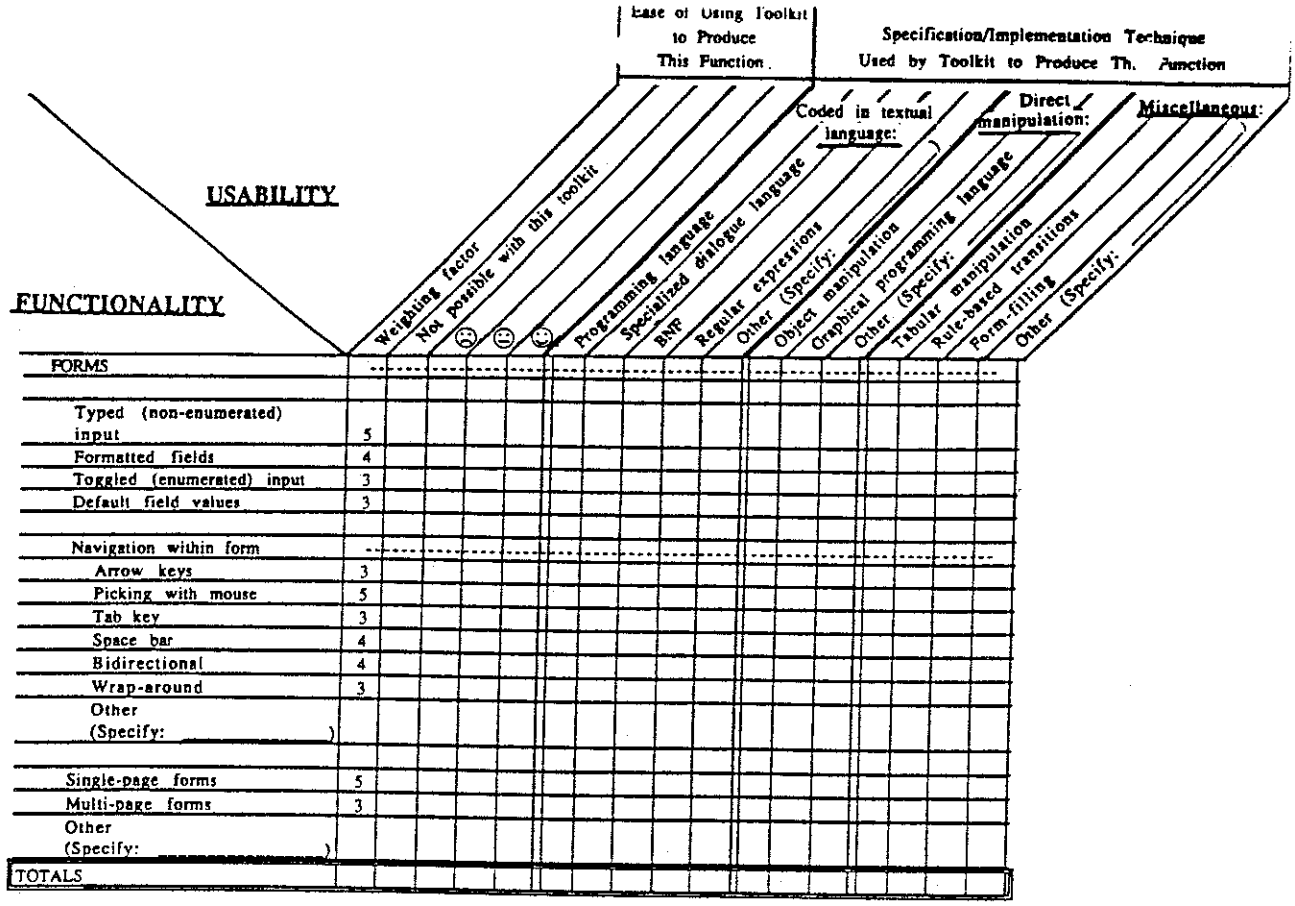

Figure 1. A Representative Page from the Evaluation Form

2.2. Results from Using the Evaluation Procedure

Several numeric results are calculated for each tool evaluated using this procedure, including:

- *Functionality rating* -- indicator of the number of interface functions the tool supports; calculated as the percentage of the total number of functions on the evaluation form that are possible with this tool;
- *Usability rating* -- indicator of the ease with which functions can be produced with the tool; calculated by considering *only* those functions possible with this tool, and assigning 1 point for a frowning face, 2 points for a bland face, and 3 points for a smiling face, this rating is the average rating earned expressed as a percentage of the highest possible usability rating; and
- *Specification/implementation technique rating* -- indicator of the degree to which a technique is used in the tool for producing possible functions; calculated as the percentage of use of a specific technique relative to all possible techniques.

A completed evaluation report contains several parts:

- *General description* of the tool being evaluated;
- Information about *sources* used in preparing the evaluation;
- *Executive summary* of overall functionality and usability ratings for the three sections (types of interfaces, types of support, general characteristics) of the evaluation form;
- *Detailed evaluation* of functionality and usability dimensions for the three sections, used to compute overall functionality and usability ratings; and
- *Glossary* defining every item in the form.

3. EMPIRICAL TESTING OF THE EVALUATION PROCEDURE

3.1. Experimental Procedure

We wished to empirically determine the extent to which different evaluators, using this evaluation procedure, would produce similar results for the same tool -- i.e., the reliability (consistency) of the procedure across different evaluators. We therefore conducted an experiment using six graduate students from Virginia Tech as research participants, all with substantial system and interface development experience. The three tools that were evaluated were:

- Bricklin's *Demo* V1,
- *HyperCard* V1.2.1, and
- SmethersBarnes *Prototyper* V1.

These tools were chosen because they support development of different application types, and we therefore assumed that they have different functionality and usability. Demo runs on IBM-style hardware and its primary use is for developing sequential or non-direct manipulation interfaces. Prototyper runs on the Macintosh and is used for developing Macintosh-style direct manipulation interfaces. HyperCard, which also runs on the Macintosh, is not strictly an interface development tool, but was chosen for this study because of its prevalence and because it has a fair number of features that support human-computer interface development.

Two different tools were evaluated by each participant, with appropriate counter-balancing, so that each tool was evaluated by four different participants. While results of evaluating the tools are inherently interesting, we were more concerned with evaluating the evaluation form and procedure -- their usability, completeness, understandability, and so on.

The experiment was conducted in three stages. During Stage 1, participants received software and manuals for the two tools they were to evaluate and learned each tool to a "reasonable" level of expertise, at their own pace, over a two week period.

In Stage 2, participants performed some baseline tasks, to insure that each participant had achieved at least a common minimum level of expertise with each tool. These tasks took no more than fifteen minutes to complete, and all participants satisfactorily performed the baseline tasks on their first attempt.

In Stage 3, participants completed an evaluation form for each of their two tools, after meeting with the experimenter who gave them an explanation of the functionality and usability dimensions and helped them complete several items on the form. During form completion, done at their own pace, participants could use any documentation and the tool itself as desired. Finally, overall functionality and usability ratings were computed, as well as specification/ implementation technique ratings, using the electronic spreadsheet.

3.2. Results of Evaluation of Each Tool

We will present two kinds of results: numbers from use of the evaluation procedure for evaluating each tool are given in this section, and results of testing the reliability of the procedure based on these numbers are given in section 3.3.

Participants reported spending six to ten hours learning a tool, and another one to two hours completing each evaluation form. Table 1 shows (unweighted) mean summary functionality and usability percentages for each tool for each of the three sections of the form, averaged from ratings of all four participants who evaluated each tool.

	DEMO		HYPERCARD		PROTOTYPER	
	Funct.	Usab.	Funct.	Usab.	Funct.	Usab.
TYPES OF INTERFACES	52	82	70	87	46	93
TYPES OF SUPPORT	56	65	76	82	38	74
GENERAL CHARACTERISTICS	28	61	50	83	31	75

Table 1. Mean Summary Results (%) of Functionality and Usability Ratings for All Three Sections of the Form

	No. of Items	DEMO		HYPERCARD		PROTOTYPER	
		Funct.	Usab.	Funct.	Usab.	Funct.	Usab.
INTERACTION STYLES:							
Menus	14	82	78	84	82	66	93
Forms	12	73	76	90	97	54	93
Typed input strings	4	19	100	69	68	19	100
Windows	3	59	92	75	81	83	91
FEATURES OF INTERFACES	27	46	71	84	92	37	84
HARDWARE DEVICES:							
Input devices	17	18	89	20	99	12	100
Output devices	7	71	78	64	85	53	92

Table 2. Types of Interfaces the Tools Can Produce: Mean Summary Results (%) of Functionality and Usability Ratings

These results mean that, for example, HyperCard can produce 70% of all types of interfaces on the evaluation form. For producing those types of interfaces, its usability rating is 87%.

Table 2, giving a more detailed breakdown of line 1 of Table 1, shows mean summary percentages for categories in the "types of interfaces" section.

To help the reader understand these numbers, consider only the "Demo" functionality and usability columns in Table 2. There are 14 kinds of menus on the detailed functionality list of the evaluation form. Demo can produce 82% of them, and can produce 73% of the 12 types of forms. This means that Demo supports a good variety of types of menus and forms. As already discussed (Section 2.2), usability is evaluated only on functions that are possible (e.g., the 82% of types of menus). Usability ratings of 78% and 76% mean Demo is reasonably easy to use to produce menus and forms, respectively. Demo can produce interfaces with only a few input devices (functionality of 18%), while output devices are much better supported (71%). Those input and output devices that Demo supports are reasonably easy to incorporate into interfaces produced using Demo (usability of 89% and 78%, respectively). Usability ratings for Demo across all categories are fairly high (ranging from 71% to 100%), indicating that Demo is a rather easy tool to use. It is important to remember that all usability ratings refer to ease of using the tool to produce an application interface, and are *not in any way* related to ease of use of that interface.

Now consider all columns in Table 2, comparing results across all three tools. Some spot comparisons show that HyperCard (with functionality of 90%) produces more types of forms than Demo (73%) or Prototyper (54%). Typed input strings are not well supported by either Demo or Prototyper (both with functionalities of 19%). HyperCard supports more features of interfaces (84%) than does Demo (46%) or Prototyper (37%). All three tools provide little support for input devices, while Demo supports more output devices (71%) than does HyperCard (64%) or Prototyper (53%). Usability ratings are generally high for all three tools. Results in Table 3, showing types of support the tools provide, can be interpreted similarly.

Results of specification/implementation techniques (details omitted due to space limitations) used by each tool indicated that HyperCard uses mostly object manipulation (80%) with some tabular manipulation (11%). Prototyper also uses mostly object manipulation (78%) and some form-filling (14%). Demo uses a mix of object and tabular manipulation, rule-based transitions, and form-filling. None of these tools uses textual languages to any extent.

A series of analyses of variance was run on the functionality and usability results to investigate which of these mean results are far enough apart to be significantly different. The main comparisons were computed across the three different sections of the form (sections shown in Table 1).

	No. of items	DEMO		HYPERCARD		PROTOTYPER	
		Funct.	Usab.	Funct.	Usab.	Funct.	Usab.
Rapid prototyping	9	75	72	86	83	58	89
Development methodology	6	58	76	84	87	50	75
Constructional model of human-computer interaction	3	75	61	78	83	25	67
Evaluation of target system interface	7	39	74	61	60	0	n/a
Database management system	2	0	n/a	75	55	0	n/a
Interface libraries	1	50	50	100	84	50	50
Help for using tool	1	100	67	100	100	75	67
Documentation of tool itself	1	100	42	100	75	100	75
Context of definition	3	58	75	84	95	25	84
Automated project management within tool	5	0	n/a	10	100	0	n/a

Table 3. Types of Support the Tools Provide: Mean Summary Results (%) of Functionality and Usability Ratings

3.3. Results of Reliability Testing

Results from use of the evaluation procedure were analyzed to determine whether numbers produced by the procedure are reliable. Reliability is a measure of consistency across different evaluators. Cronbach's alpha, a measure of internal consistency, was computed for each category and for the entire form. The greater the value of alpha (which has an upper limit of 1.0), the higher the reliability. However, alpha values of .50 or greater are considered reasonable. Alpha values for functionality ratings only are shown in Table 4.

Alpha values varying from negative to 1.0 were also found when alpha values were computed for usability ratings. Most alpha values for various categories clustered around .50. Across all functionality items in the form, the alpha values for usability ratings were .53, .52, and .61 for Demo, HyperCard, and Prototyper, respectively.

4. DISCUSSION

4.1. Tool Evaluation Results

The analysis of variance showed that HyperCard has a significantly higher functionality for types of interfaces a tool can produce, and Demo has about the same functionality as Prototyper (line 1 of Table 1, and Table 2). These results are not surprising, since HyperCard is a more general purpose tool. The analysis of variance showed that the only significant difference for usability ratings for types of interfaces a tool can produce is between Prototyper and Demo (line 1 of Table 1, and Table 2); Prototyper is indeed more usable than Demo but the difference in usability between HyperCard and either of the other two systems is not significant. These are again reasonable results: Prototyper runs on a Macintosh and might therefore be expected to be easier to use. Although HyperCard also runs on a Macintosh, its greater functionality may actually interfere with its usability.

The analysis of variance showed that all three systems are indeed significantly different in their functionality for types of support, with HyperCard providing the most types of support (line 2 of Table 1, and Table 3), again expected because of its generality, followed by Demo and then Prototyper. The analysis of variance showed that for usability, the only significant difference is between the highest usability (HyperCard) and the lowest (Demo); Prototyper, in between, is not significantly different in usability for types of support than either of the other two systems.

While the results in Table 1 (line 3) indicate that HyperCard has more general tool characteristics than Prototyper, which has more than Demo, the

	DEMO	HYPERCARD	PROTOTYPER
INTERACTION STYLES:			
Menus	.53	.71	.63
Forms	.58	.58	.74
Typed input strings	.49	0	.89
Windows		.92	-2.67
undefined			
FEATURES OF INTERFACES	.65	.43	.52
HARDWARE DEVICES:			
Input devices	1.0	.86	1.0
Output devices	1.0	.55	.79
TYPES OF SUPPORT	.73	.65	.81
GENERAL CHARACTERISTICS OF TOOL	-.27	0	.81
ACROSS ENTIRE FORM	.79	.70	.76

Table 4. Cronbach's Alpha Values for Functionality for Each Category of the Form and Across the Entire Form

analysis of variance showed that the three tools in fact do not differ significantly on either functionality or usability of general characteristics.

More complete comparison of the tools, say, to choose one for a development environment, would necessitate a look into details of individual items on the evaluation form, from which these overall results were calculated.

4.2. Reliability Testing Results

The value of Cronbach's alpha varied considerably across different categories of the evaluation form, for both functionality ratings and usability ratings. Several categories had high alpha values for the functionality ratings across all three tools (see Table 4), in particular, input/output devices and types of support, indicating that results for these categories are more reliable than those categories with low alpha values. Unusual alpha values (i.e., zero, negative, and undefined) for functionality ratings for typed input strings, windows, and general characteristics suggest that Cronbach's alpha may not be the most appropriate measure of reliability for categories with only a few (3 or 4) items. Computing alpha values across all 127 functionality items in the form, overall reliability for functionality ratings were found to be quite respectable, ranging from .70 to .79, as shown in the bottom line of Table 4.

Alpha values computed for usability ratings were somewhat lower and more varied than those for functionality ratings. This was expected, since usability is inherently more subjective than functionality, and therefore more likely to be inconsistent across participants. Also, the evaluation form's usability rating has three choices (the three faces), rather than the two choices (possible or not possible) for functionality, thereby increasing the potential for disagreement among participants. Categories with high alpha values for usability (e.g., hardware devices) were less open to interpretation. Despite these limitations, the values of alpha for usability ratings over all items in the form were all greater than .50, again indicating reasonable reliability, and therefore consistency for the evaluation form. Overall, these alpha values indicate that any two or more expert evaluators using this evaluation form should get very similar results for a tool's evaluation.

Qualitatively, participants said in interviews that they felt results fairly represented tool capabilities. As a result, rather than producing *ad hoc* evaluations, they felt the form provides a structured, consistent instrument for evaluating and comparing tools, as well as presenting results of those evaluations. These comments indicated participants' positive feels toward the procedure, despite the fairly lengthy time (at least 20 hours each) they had volunteered.

5. CONCLUSIONS AND FUTURE WORK

Results across different participants (evaluators) were substantially more similar than we expected. Since this was the first time the evaluation procedure had been formally used, we anticipated the possibility of widely varying results across different participants for the same tool. However, results of the empirical testing indicate that our procedure provides reasonably reliable, consistent results across different evaluators.

Examination of details across different participants revealed several instances where one participant marked a function (e.g., typed input strings) as "not possible" while another marked it otherwise. Investigation showed this was due, as expected, to two common reasons:

- Differences in interpretation of glossary definitions, and
- Differences in expertise level of different evaluators using the same tool.

For example, some participants found a way to produce a particular function, while others did not. Interestingly, most discrepancies were found either in rather obscure, poorly understood items or in items not explicitly supported by a tool, for example, producing "typed input strings" using Prototyper and Demo. When they were not explicitly possible, some participants used intuition and creativity to produce typed input strings with a tool. These two differences suggest that detail in the glossary could be clarified, and formal evaluator training should be explored.

In general, we believe that this procedure cannot be effectively used by a non-expert evaluator; it should be used only by one who has thoroughly learned the tool(s) to be evaluated and is familiar with the procedure and form. This is to be expected: given the complexity of the kinds of tools we are attempting to evaluate, a procedure for evaluation is itself lengthy and complex. An evaluator could easily spend a week or more becoming thoroughly familiar with a tool to be evaluated. Results of the reliability testing indicate that two or more expert evaluators should get the same overall results when using the evaluation form, and should therefore reach the same conclusions about tools evaluated using this form.

The procedure's focus on quantitative results can be misconstrued. In particular, the overall (summary) ratings give only macro results; an evaluator *must* go to the detailed sections of the form in order to determine specific information about a tool. The numeric ratings simply provide guidelines for evaluating and comparing tools; they do not give absolute answers. They serve as an instrument to aid in decision-making, but do not themselves mandate any particular decision.

Finally, there are great advantages to a "checklist" evaluation procedure such as we have described. It encourages its users (the tool evaluators) to broaden their thinking about a tool being evaluated, by presenting them with a structured and wide variety of possible choices, many of which they might not think of on their own. However, this very structure can be limiting, simply because it is fixed at any point in time, and thus can actually narrow the set of questions asked about a particular tool. As an initial response to this, we have made the form extensible by allowing evaluators to add items to it when appropriate (in Figure 1, note spaces on the form for "other"). We are also investigating including an initial phase in the evaluation procedure in which evaluators must determine, in some structured fashion, a *desired*, context-sensitive set of functions for a tool within their particular environment, *before* proceeding with evaluation of candidate tools using our procedure. This would help debias the evaluator and remove some of the limitations associated with a checklist approach.

Several other open issues are actively being pursued. For categories with low alphas, we are investigating how to improve reliabilities. Determining validity (i.e., whether the procedure measures what we intend it to measure) of the procedure is also being investigated; this study addressed only reliability. Validity is even more difficult because of a lack of expert comparisons for cross-validation purposes. Without a comparable technique for evaluating tools, which currently does not exist, validity cannot be effectively assessed. Customization for a particular development environment and decreasing the length and complexity of the form are also being studied.

6. SUMMARY

To our knowledge, this evaluation procedure is the first attempt at developing a standardized technique for evaluating and comparing human-computer interface development tools. Contributions of this research include:

- Method for systematically and consistently evaluating all aspects of a tool;
- Concept of quantitative functionality and usability ratings for a tool;
- Taxonomy of types of interfaces that can be produced with a tool; and
- Identification of specification/ implementation techniques used by a tool.

Our goal is that this research will result in a rigorous, trusted methodology for evaluating human-computer interface development tools. We have just begun understanding the issues, problems, and promises involved in developing such a methodology.

Acknowledgements

The author would like especially to thank graduate student Kay Tan, who conducted the reliability study, and the participants who volunteered their time for the study. Appreciation also goes to Dr. Robert S. Schulman and Dr. Bruce Koons, who gave invaluable expert advice on the experimental study. Thanks also to Jo-Anne Lee Bogner, who produced the camera-ready copy. This research was funded by the Software Productivity Consortium and the Virginia Center for Innovative Technology.

References

Arble, F. (1988). Private communication.

Borenstein, N. S. (1985). The evaluation of text editors: A critical review of the Roberts and Moran methodology based on new experiments. *Proceedings of CHI'85 Human Factors in Computing Systems*, (pp. 99-105). New York: ACM.

Cohill, A. M., Gilfoil, D. M., & Pilitsis, J. V. (1988). Measuring the utility of application software. In H. R. Hartson & D. Hix (Eds.), *Advances in Human-Computer Interaction*. (Vol. 2). Norwood, NJ: Ablex Publishing Corp.

Roberts, T. L., & Moran, T. P. (1983). The evaluation of text editors: Methodology and empirical results. *Communications of the ACM*, *26*(4), 265-283.

Totten, S. (1989). Private communication.