An Evaluation Procedure for Human-Computer Interface Development Tools

Deborah Hix and Kay Tan

TR 88-44

AN EVALUATION PROCEDURE FOR HUMAN-COMPUTER INTERFACE DEVELOPMENT TOOLS

Deborah Hix*
Department of Computer Science

Kay Tan
Department of Industrial Engineering and Operations Research

Virginia Polytechnic Institute & State University
Blacksburg, VA 24061 USA
Phone: 703/961-6199
e-mail: hix@vtopus.vt.cs.edu

ABSTRACT: Human-computer interface development tools -- often called user interface management systems or UIMS -- are interactive systems that support production and execution of the human-computer interface. Despite their proliferation, no method exists for their systematic evaluation or comparison. We have developed an evaluation procedure that uses a standardized technique to produce quantifiable criteria for evaluating and comparing human-computer interface development tools. The procedure produces ratings along two dimensions: functionality and usability. Specification/implementation techniques used by the tool are also quantitatively rated. An empirical study indicates that the procedure produces reliable results. The procedure is already being used in one commercial environment.

KEYWORDS: Human-Computer Interface Development Tools, Functionality, Usability, Evaluation, User Interface Management Systems, UIMS

SUGGESTED TOPIC AREA: User Interface Management Systems or User Interface Tool Kits

NUMBER OF WORDS: 2952**

---

* Co-author for correspondence

** Includes references and acknowledgement, but not figures: Clayton Lewis said that figures are excluded from the word count

1. INTRODUCTION

1.1. Motivation for a Tool Evaluation Procedure

Tools for human-computer interface development -- often called user interface management systems or UIMS -- conjure up visions of icons and images, windows and words, mice and other media that comprise the human-computer interface of an interactive system. Human-computer interface development tools are themselves interactive systems, ones that support production and execution of the human-computer interface. With their proliferation, evaluations and comparisons of such tools are often done, but without a formal, structured approach. State-of-the-art in tool evaluation is based mostly on subjective opinions. These tools are difficult to evaluate because of their many different varieties, their relative newness, and their inherent complexity.

We have developed and empirically validated the reliability of an evaluation procedure that uses a standardized technique to produce quantifiable criteria for evaluating and comparing human-computer interface development tools. These data could be used, for example, to choose 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 evaluation of such tools. Our procedure is already being used in a commercial testbed environment at GTE Data Systems in Tampa, Florida [Arble 1988].

1.2. Related Work

Our evaluation procedure is similar in approach to the Roberts and Moran [1983] methodology for evaluating text editors. The basis of their approach was classification of editing tasks and evaluation along several dimensions, including time to perform tasks, error costs, learning time, and functionality. Replication studies 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 for evaluating software packages, particularly commercial systems. It was used at AT&T to select a word processing package for an engineering group. Criteria such as performance, documentation, and support were evaluated. Such research provided ideas from which we developed our evaluation procedure.

2. OVERVIEW OF THE TOOL EVALUATION PROCEDURE

2.1. Description of the Evaluation Procedure

Because our procedure [Hix and Tan 1988] revolves around "hands-on" use of the tool to be evaluated, the tool must first be acquired. As shown in Figure 1, after learning the tool the evaluator completes a detailed 28 page form that is organized around two dimensions:

- Tool functionality, and
- Tool usability.

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

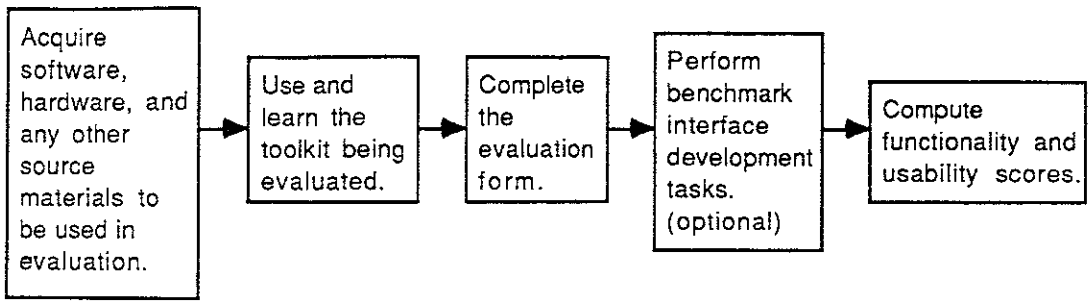

FIGURE 1. High-Level Block Diagram of Steps in the Evaluation Procedure

After completing the detailed portion of the form, the evaluator calculates overall functionality and usability ratings (using an electronic spreadsheet) for an executive summary. If desired, the evaluator performs benchmark interface development tasks.

The functionality dimension of a tool's evaluation contains three sections:

- *Types of interfaces* a tool can support, including *interaction styles* (e.g., menus, windows); *features of interfaces* (e.g., types of navigation, defaults, graphics); and a variety of *hardware (I/O) devices;*
- *Types of support* provided for the general process of interface development (e.g., rapid prototyping, evaluation, libraries, documentation); and
- *General characteristics* of a tool (e.g., consistency, integration).

The usability dimension 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 to use (indicated by a frowning face), adequate to use (bland face), or easy to use (smiling face) to produce that function. We attempted to reduce individual interpretation by operationally defining each "face."
- *Objective evaluation* measures human performance using suggested benchmark interface tasks that are customizable to a specific working environment (this aspect has not been fully developed and will not be addressed here).

For those types of interfaces that are possible with the tool, the evaluator indicates the primary type of specification/implementation technique the tool uses, namely:

- *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 2 shows a representative page from the "types of interfaces" section, the page for evaluating "forms."

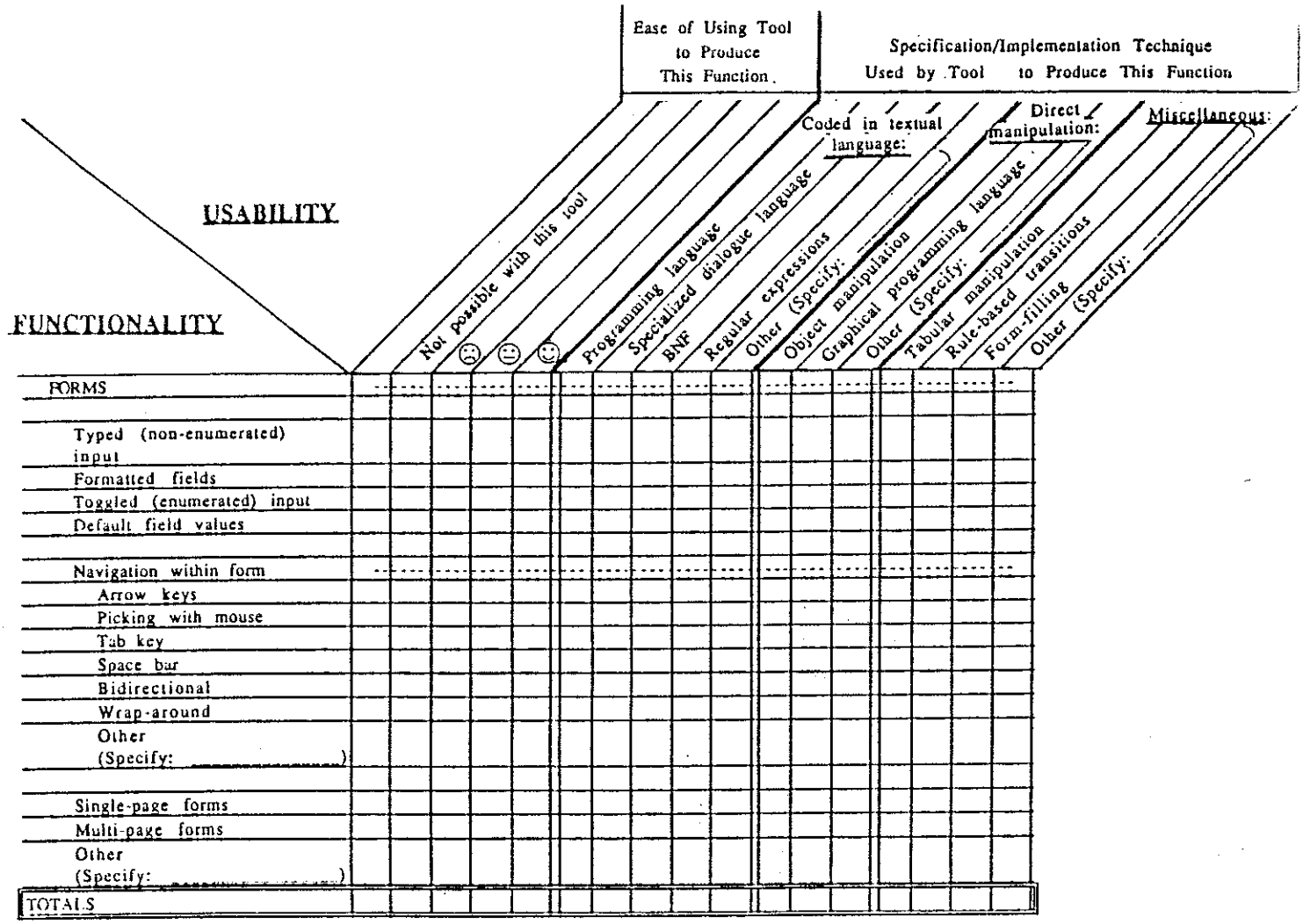

FIGURE 2. A Page from the Evaluation Form

2.2. Results from Using the Evaluation Procedure

Numeric results computed for each evaluated tool include:

- *Functionality rating* -- indicator of 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 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 for bland, and 3 for smiling, this rating is the average score earned expressed as a percentage of the maximum score possible; and
- *Specification/implementation technique rating* -- indicator of the degree to which a technique is used in the tool.

A completed evaluation report contains several parts:

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

3. EMPIRICAL VALIDATION OF THE EVALUATION PROCEDURE

3.1. Participants, Materials, and Procedures

To test the reliability of the evaluation procedure, we conducted an experiment using six graduate students from Virginia Tech as research participants (subjects). All had substantial system development experience. Three tools representing different application types (and presumably different functionalities and usabilities) were evaluated:

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

Each participant evaluated two different tools, with appropriate counter-balancing, so that each tool was evaluated by four different participants. While evaluations of the tools are inherently interesting, we were more concerned with examining the evaluation procedure itself.

The study was conducted in three stages. During Stage 1, participants received software and manuals for the two tools they were to evaluate. They were asked to learn each tool during the next two weeks. Participants were told to contact the experimenter when they felt they had gained sufficient proficiency to perform baseline tasks with each tool. These tasks were designed to ensure that each participant attained at least a common baseline level of expertise with each tool, to reduce between-participant variance. Participants were also given the evaluation form to inspect, but were told not to complete it until after they demonstrated their expertise through the baseline tasks.

During Stage 2, participants performed the baseline tasks. Our expectation was that these tasks, performed by an experienced evaluator after six to eight hours spent learning a tool, should take no more than fifteen minutes to complete. All participants satisfactorily performed the baseline tasks on their first attempt.

During Stage 3, participants completed the evaluation form for both tools. Participants were first given an explanation of functionality and usability dimensions and use of the glossary, and were assisted in completion of several items on the form. They then completed the form for each tool at their own pace, using any documentation and the tool itself as desired.. For any term they did not understand even after looking in the glossary, participants could contact the experimenter for clarification. Finally, the experimenter computed overall functionality and usability ratings.

3.2. Results of Using the Procedure to Evaluate the Tools

Two kinds of results are presented here: evaluation results for each tool (in this section), and reliability testing of the procedure based on these results (in Section 3.3).

Participants reported spending six to ten hours learning a tool, and another one to two hours completing each evaluation form. Tables 1, 2, and 3 show mean summary percentages for each tool, averaged from ratings of all four participants who evaluated each tool. (Due to space limitations, individual participant ratings [Hix and Tan 1988] are not shown.)

	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 DIALOGUE	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 1. Types of Interfaces the Tools Can Produce: Mean Summary Results (%) of Functionality and Usability Ratings

To help the reader understand these numbers, consider only the "Demo" functionality and usability columns in Table 1. Of the 14 types of menus on the form's detailed functionality list, Demo can produce 82%, and of the 12 types of forms, 73%. This means Demo supports a variety of menus and forms. Usability is evaluated only for possible functions (e.g., the 82% 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%); both input and output devices are reasonably easy to incorporate into interfaces produced using Demo (usability of 89% and 78%, respectively). Across all categories, participants had fairly high usability ratings, 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 the other columns in Table 1, comparing results across all three tools. Some spot comparisons show that HyperCard produces more types of forms (functionality of 90%) than Demo (73%) or Prototyper (54%). Typed string inputs are not well supported by Demo or Prototyper (both functionalities of 19%). HyperCard supports more features of dialogue (84%) than does Demo (46%) or Prototyper (37%). All three tools provide about the same 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 2, showing types of support the tools provide, can be interpreted similarly.

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

Specification/implementation techniques used by each tool are shown in Table 3. 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 the tools uses textual languages to any extent.

	DEMO	HYPERCARD	PROTOTYPER
CODED IN TEXTUAL LANGUAGE			
Programming language	-	8	1
Specalized dialogue language	5	1	-
BNF	-	-	-
Regular expressions	-	-	-
Other (Specify:)	-	-	-
DIRECT MANIPULATION			
Object manipulation	17	80	78
Graphical programming language	-	-	3
Other (Specify:)	-	-	-
MISCELLANEOUS			
Tabular manipulation	32	11	2
Rule-based transitions	28	-	-
Form-filling	18	-	14
Other (Specify: plug-in)	-	-	2

TABLE 3. Specification/Implementation Techniques Used By the Tools: Mean Summary Results (%)

3.3. Results of Testing the Procedure for Reliability

Results of the evaluations were statistically analyzed to determine whether numbers produced by the procedure were reliable. Reliability is a measure of consistency across evaluators. Cronbach's alpha, a measure of internal consistency, was computed (for functionality only) for each category and for the entire form, with values shown in Table 4. The greater the value of alpha (which has an upper limit of 1.0), the higher the reliability.

	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 DIALOGUE	.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 Each Category of the Form and Across Entire Form

4. DISCUSSION

4.1. Evaluating the Tools

At the highest level, these results (Table 1) show that HyperCard has the greatest functionality for types of interfaces a tool can produce, and Demo has slightly more functionality than Prototyper. This is expected, since HyperCard is a general purpose tool. Prototyper is more usable than Demo or HyperCard, which both have about the same usability for types of interfaces a tool can produce. This again is reasonable: Prototyper runs on a Macintosh and would therefore be expected to be easier to use. Although HyperCard also runs on a Macintosh, the complexity due to its greater functionality may interfere with its usability. HyperCard provides the most types of support (Table 2), again expected because of its generality, followed by Demo and then Prototyper, and also has the highest usability for these types of support. Direct manipulation was the primary specification/implementation technique (Table 3) for HyperCard and Prototyper, which was undoubtedly reflected in their overall high usabilities. A more complete comparison of the tools, say, to choose one for a development environment, necessitates inspection of individual items on the form, rather than just overall ratings.

4.2. Testing the Procedure for Reliability

Evaluation ratings are of questionable usefulness unless we can show they are reliable; that is, will different evaluators produce similar results for the same tool? The value of Cronbach's alpha varied considerably across categories of the evaluation form. Several categories had high

alphas, in particular, input/output devices and types of support, indicating that ratings for these categories are more reliable than categories with low alphas. Unusual values (i.e., zero, negative, and undefined) for typed input strings, windows, and general characteristics suggest that Cronbach's alpha may be an inappropriate measure of reliability for categories with only a few (3 or 4) items.

As can be determined from the detailed evaluation (not shown here), some of the ratings for the same tool had a wide range between participants. This was not surprising; we found that participants did have a common minimal level of expertise with the tools (demonstrated by the baseline tasks), but the upper limit of their expertise varied. Participants with more computer experience learned more about each tool than those with less experience. Computing alpha across all 127 functionality items on the form, overall reliability ratings were respectable, ranging from .70 to .79.

Qualitatively, participants said in exit interviews that they felt results fairly represented tool capabilities. As a result, rather than producing *ad hoc* evaluations, they felt the form provided a structured, consistent instrument for evaluating and comparing tools. These comments indicated participants' positive feelings toward the procedure, despite the fairly lengthy time (about 20 hours each) required.

5. CONCLUSIONS AND FUTURE WORK

Between-participant results were substantially more similar than we expected. We anticipated the possibility of widely varying results across different participants for the same tool, since this was the first time the evaluation procedure had been used. However, results indicate that our procedure provides reasonably reliable results across different evaluators.

Examination of between-participant details 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 variances were found in categories not explicitly supported by a tool, for example, producing "typed input strings" using Prototyper and Demo. Because they were not explicitly possible, some participants used intuition and creativity to produce "typed input strings" using a tool. These differences suggests that detail in the glossary could be clarified, and formal evaluator training should be explored. In general, the inference is that this procedure cannot be reliably 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 to be evaluated, a procedure for evaluation will necessarily be lengthy and complex.

Several open issues are actively being pursued. For categories with low alphas, we are investigating ways to improve reliabilities. Determining validity 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. Decreasing the length and complexity of the form is also being studied.

6. SUMMARY

This evaluation procedure is, to our knowledge, 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 authors would like especially to thank the participants who gave their time for this study. Our appreciation also to Dr. Robert Schulman and Bruce Koons, who gave invaluable expert advice on experimental design and analysis of results. 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., pp. 128--158.

Hix, D. and Tan, K. (1988). *A procedure for evaluating human-computer interface development tools: Its use and validation.* Computer Science Department Technical Report, Virginia Tech, Blacksburg VA 24061.

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