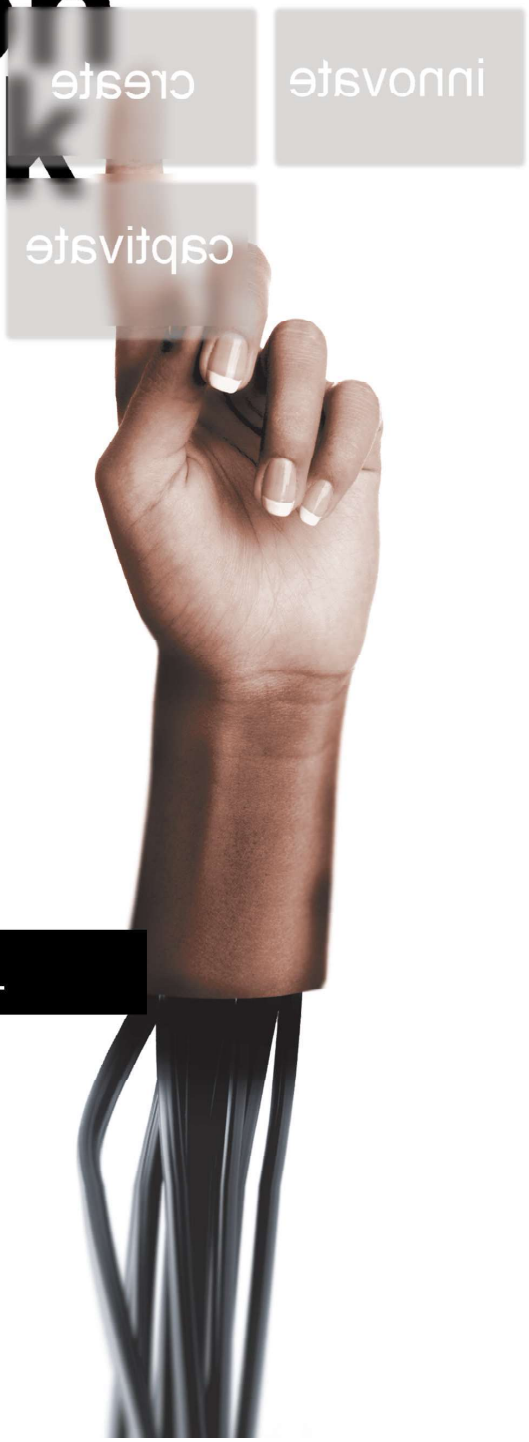
The Human–Computer Interaction Handbook

Fundamentals, Evolving Technologies, and Emerging Applications

Third Edition



Edited by Julie A. Jacko, Ph.D.

CRC Press
Taylor & Francis Group

56 Inspection-Based Evaluations

Gilbert Cockton, Alan Woolrych, Kasper Hornbæk, and Erik Frøkjær

CONTENTS

Usability inspection methods (UIMs) are approaches to usability evaluation based on expert inspection of a user interface and the probable user interactions with it. They can be applied to any designed artifact during development: a paper prototype, a storyboard, a working prototype (e.g., in Macromedia Flash™ or in Microsoft PowerPoint™), tested production software, or an installed public release. They are *analytical* evaluation methods, which involve no typical end users, unlike empirical methods such as user testing. UIMs require only availability of a designed artifact, trained evaluators, and supplementary project/evaluator resources. The resource requirement for evaluation is thus low: UIMs were one of the first groups of *discount methods* within human–computer interaction (HCI). Their origins as discount methods are important. Their inventors focused on reducing the cost of usability evaluation to a level that they judged to be acceptable for software development projects in the early 1990s. This focus on cost was at the expense of critical reflection and systematic scientific evaluations. Perversely for a user-centered field such as HCI, there was limited consideration of how evaluators would actually use methods, both *cognitively* from a perspective of problem-solving behaviors, strategies and tactics, and *socially* from a perspective of usability evaluation as *work* within project *contexts*. More recently, HCI has focused more on the affective aspects of interaction, and these too now need to be considered when assessing UIMs. It is not enough for UIMs to be better assessed and supported by better advice on their use by evaluators within usability work contexts. UIMs must also "feel right" to evaluators. It is important that evaluators believe that UIMs are helping them to find important usability problems and to understand and apply them well enough to be able to recommend effective design changes to remove the problems. It is unrealistic to expect UIMs alone to guarantee high quality evaluation. Interaction design is a complex challenging activity that demands extensive expertise and understanding.

This third version of this chapter takes a more critical perspective than previous ones, drawing on recent research in the context of 4 years of collaboration between researchers from over 20 European countries (the MAUSE project on Maturity of Usability Evaluation, www.cost294.org). At the heart of the problem with unrealistic expectations for UIMs is the word "method" itself, which hides many pitfalls

for the unwary. Clearly, no published generic method can be complete. For example, accounts of user testing "methods" may give guidance on recruiting and screening test users and on designing and facilitating test tasks, but these accounts will never tell you who to recruit or which criteria to apply when screening potential test users, nor will they tell you exactly which tasks to facilitate. No account of user testing could possibly do this, because different interactive systems are designed to support different tasks for different groups of users to different standards of work or other qualities. While this all should be obvious, it is worth reminding anyone new to a design or evaluation method that practitioners and researchers *must work to get methods to work*. Methods do not apply themselves, nor do they provide all the resources (e.g., project-specific participant screening criteria) required to apply them.

Methods only exist in the past tense: you know what your method *was*, but at the outset of usability work, you are highly unlikely to know what your method *will be*. At best you will have a firm outline, supported by extensive project-specific details, of how you will *approach* usability work. We prefer to use the word *approach* to "method." In much commercial work, clients ask about a potential supplier's overall approach, rather than for step-by-step details of the methods that they will use. We believe that "approach" is a more accurate term for project-independent work structures, that is, prefigured activity skeletons that must be configured for specific projects. As with all human activities, methods are achievements and can never be premonitions. Evaluators need to see through the commodification of resource bundles as "methods," and be realistic about their real value prior to configuration and augmentation for specific evaluation contexts.

This chapter introduces UIMs, presents several example methods, and highlights the limited resources provided by each. Risks clearly arise from these limited resources, but well-informed practices disproportionately improve evaluation performance, improving cost-benefit ratios. Well-informed usability practices rely on additional resources provided by specific project environments, including evaluators themselves. Thus, evaluators must complement UIM resources with their own. As a result, it is very hard to rank UIMs or to make firm recommendations for the superiority of one UIM over another, even in fairly specific circumstances. The reason for this is that actual performance with a UIM depends on how it is configured and used in practice. This chapter, therefore, focuses on what is required to turn the rough approaches of UIMs into effective methods.

56.1 DEFINITIONS AND CONTRASTS

A UIM is an analytic approach to usability evaluation based on expert inspection of a user interface and the probable user interactions with it. A UIM *provides* resources that can be *applied directly* to an interaction design artifact, and/or probable user interactions with that artifact, and does not require end user resources. A resource is any element of an evaluation method, for example, knowledge about interaction design, evaluation procedures, problem report formats, checklists, or problem severity criteria. *Direct application* contrasts UIMs with model-based methods, which are *indirectly* applied via design representations (or models), requiring construction of models and secondary application of analyses to designed artifacts. Using the GOMS (Goals, Operators, Methods and Selection rules) method, for example, a task model would be analyzed and the results would have to be reframed to address the actual design artifact. In contrast, UIMs such as heuristic evaluation (HE) directly identify design features that may cause user difficulties. UIMs require fewer resources than model-based methods and thus commit evaluators to less work. While the reduction in costs could be accompanied by a reduction in benefits, the increase in costs for model-based methods may not be justified by a commensurate increase in benefits. In short, there is no fixed connection between the resources provided by evaluation approaches and their cost-benefit ratios. Evaluators thus need to pay careful attention to planning, supported by continuous critical reflection on the effectiveness of specific usability work practices. We next briefly review three UIMs to informally identify resources associated with, and missing from, each.

56.2 THREE EXAMPLE USABILITY INSPECTION METHODS

56.2.1 Heuristic Evaluation

With HE (Nielsen and Molich 1990; Nielsen 1992, 1994b), evaluators inspect a system with a view to discovering breaches of heuristics, which are "rules of thumb," rather than focused usability guidelines or exact style conformance rules. Breaches of heuristics indicate possible usability problems. One example of a heuristic breach is when a file is downloaded from a website, or intranet, without visible feedback to inform them of download progress. The failure to provide feedback breaches Nielsen's *visibility of system status* heuristic.

Heuristics are HE's main resource. The 10 current heuristics (Figure 56.1) were derived from analysis of 249 known usability problems, which had been established from evaluation of

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

FIGURE 56.1 HE's heuristics. (From Nielsen, J. 1994. *Usability Inspection Methods*. John Wiley & Sons. With permission.)

11 different interactive systems (Nielsen 1994b). The objective was to produce a compact heuristics set that best covered a set of known usability problems. HE also prescribes a specific procedure for performing an evaluation. Nielsen (1994b) recommends that analysts go through the interface at least twice. The first "pass" allows the evaluator to get a "feel" for the system, that is, both the general scope of the system and the flow of interaction. In the second and subsequent passes, the evaluator can focus on specific elements of the user interface. This procedure is HE's second resource, but advice is not given on how to structure the first pass or on how to focus in the second pass and associate possible usability problems with breached heuristics. These two critical resources must be provided by evaluators and/or their project/organizational context.

56.2.2 Cognitive Walkthrough

Cognitive walkthrough (CW) (Lewis and Wharton 1997) is a UIM primarily concerned with learnability, initially developed to assess "walk up and use systems" and later extended to task-based assessment of more complex systems. In CW, evaluators first of all assess if the user can form appropriate goals (Question 1) and then assess if the user can choose the appropriate actions to achieve their goal (Questions 2–4). The four CW questions are the following:

1. Will the users try to achieve the right effect? Does the user know what to do? Is it the correct action?
2. Will the user notice that the correct action is available? Is the action visible? Will users recognize it?
3. Will the user associate the correct action with the effect to be achieved? The action may be visible, but will the user understand it?
4. If the correct action is performed, will the user see that progress is being made toward solution of the task? Is there system feedback to inform the user of progress? Will they see it? Will they understand it?

The four questions, and an associated outline walkthrough procedure, are CW's main resources. Assuming the appropriate goal has been formed, the evaluator then breaks the "goal" down into the component task steps required to successfully achieve it. At each step, the evaluator asks questions 2–4. A negative response to any of the CW questions indicates a *possible* usability problem. To identify *probable* problems, evaluators must form success or failure judgments based on the cumulative impact of possible problems for a task. CW provides no guidance on the selection of tasks or on procedures for deciding success or failure cases. As with HE, these two critical resources must be provided by other means.

56.2.3 Metaphors of Thinking

Metaphors of thinking (MOT) (Frøkjær and Hornbæk 2002, 2008; Hornbæk and Frøkjær 2004) is a psychological-based inspection technique developed explicitly to address users' thinking, in the belief that the importance of such thought affects interaction. The metaphors are intended to support both evaluators and designers in understanding the importance of human thinking in interaction and to "stimulate critical thinking." Figure 56.2 summarizes the five metaphors. The basic method of performing an MOT inspection is that evaluators define representative tasks and "walk through" the interface. Evaluators make note of any usability problems identified, typically through violating the criteria in each metaphor. The main resources provided by MOT are thus these five metaphors and the prescription for their use. As with HE and CW, all other resources must be provided from other project/evaluator sources.

Metaphor M1: Habit Formation is Like a Landscape Eroded by Water

This metaphor is based concerned with the human trait of habit forming. The example used to illustrate this metaphor is that of adaptive menus that prevent habit forming. Since the menu's position can change from when an item was previously selected, habit forming is almost impossible.

Metaphor M2: Thinking as a Stream of Thought

The benefit of this metaphor is explained by the fragility of "stream of thought" and how interruptions can severely impact on task completion time. The example to illustrate this is that concentration can be affected even by such useful tools as e-mail alerts and automatic spelling checkers that, despite their respective benefits, can be distracting.

Metaphor M3: Awareness as a Jumping Octopus

This metaphor draws attention supporting users' associations with effective means of focusing within a stable context and, like CW, prompts consideration of whether users associate interface elements with the actions and objects that they represent. Users should be able to switch flexibly between different parts of the interface.

Metaphor M4: Utterances as Splashes over Water

This metaphor addresses support for changing and incomplete utterances, prompting consideration of alternative ways of expressing the same information, the clarity of interpretations of users' input in the system, and the risks of systems making a wider interpretation of users' input than users intend or are aware of.

Metaphor M5: Knowing as a Building Site in Progress

This metaphor is based on the notion that human knowing is incomplete, so users should not be forced by the application to depend on complete or accurate knowledge or to pay special attention to technical or configuration details before beginning to work.

FIGURE 56.2 Metaphors of thinking. (Adapted from Frøkjær, E., and K. Hornbæk. 2008. *ACM Trans Comput Hum Interact* 14(4):1–33.)

56.3 USABILITY WORK AND PRACTICE

Until recently, evaluation methods have been effectively viewed as if they were very similar to task methods in model-based approaches such as GOMS (Chapter 57), that is, evaluators follow a fixed predictable sequence of task steps to

perform an evaluation. This has never been stated explicitly within evaluation method research, but many method comparisons assume that there will be a strong *method effect*, even when the *evaluator effect* (Hertzum and Jacobsen 2001) has been so clearly demonstrated and documented. In other words, comparisons have continued to assume that methods have strong effects even when it has been shown that evaluators are responsible for large variations in outcomes. Method effects require any differences in performance between evaluators to be solely due to the UIM being used. Gray and Salzman (1998) in their thorough critique of 1990s' evaluation method comparisons noted that the causal factor was more likely to be *evaluators using method X* rather than just *method X* alone. For the latter to be so, UIMs would have to be well defined and complete as to eliminate evaluator effects. Given the limited evaluation resources provided by HE, CW, and MOT, this is unlikly to be so.

The persistence of method effect assumptions in comparison studies reflects the origins of usability engineering in the *first wave HCI* of the 1980s, which was strongly influenced by cognitive psychology. The GOMS method, a first-wave HCI approach, used a very crude approximate model of expert human planning, which ignored two fundamentals of planning within HCI (Young and Simon 1987). First, the activity of planning is intimately interleaved with the execution of plans, and second, simple, partial plans are more appropriate than complex, detailed ones. Thus by the late 1980s, cognitive scientists such as Young and Simon had shown that viewing methods as rigid pre-scripted plans was wrong. Second wave HCI, with its "turn to the social" in the 1990s, reinforced these positions with rich accounts of human work that showed that plans were *resources* and not *scripts* (e.g., Suchman 1987). However, only in the last few years has HCI research on evaluation method usage finally treated usability practices as human work, with all the variations and contingencies that this involves. We now briefly review this delayed "turn to the social" in evaluation method research and next clarify its implications for our understanding of usability evaluation.

56.3.1 Realities of Usability Work

Some HCI research has continued to compare evaluation methods based on an unrealistic position on the power of methods, but usability specialists cannot base their work on such illusions. Extensive project-specific requirements (e.g., choosing participants, reporting results) distance methods in use from published methods, even when these superficially appear to be detailed step-by-step practitioners' "cookbooks." Usability work does not get by through "choosing a method"; however well such a choice can be grounded. Instead, usability work *combines* several methods (Rosenbaum 2008), which generally compensate for each other's weaknesses, reducing or even removing the relevance of rankings or assessments of isolated methods. Molich et al. (2004) showed how teams that tested the same website differed on many specific choices on how to conduct a usability test. For instance, with respect to test reporting and task selection, marked differences were found. The nine teams not surprisingly found different sets of problems. Yet, very few scientific studies look at the combination of methods (though see Uldall-Espersen, Frøkjær, and Hornbæk 2008); this supports our argument that very few comparative research studies' evaluation investigates methods as they are used in practice. Rosenbaum's (2008) account of the evolution of usability practice shows a clear move over the last decade to project-specific combinations of multiple methods. To use a culinary analogy, usability work prepares meals, not individual dishes.

Method research in HCI should not primarily focus on supporting the choice of *a* method, nor should it attempt to answer questions such as what, in some specified context, is the "best" method, where "best" may mean most productive, most thorough, most valid, easiest to use, or cheapest to use, casting usability work as a simple choice of methods. Once chosen, the expectation is that evaluators will faithfully adhere to a method. Real world choices can never be specified wholly in terms of options and selections. There is more to choice than the menu. There is also the diner and their needs. While criteria such as nut allergies have straightforward consequences for choices from menus, wanting something "interesting and different" does not. The goodness of a choice lies in the context of the choice. Thus, Furniss (2008) argues that an appropriate choice and use of a method is functionally coupled to the project context, including client biases, practitioner expertise, their relationship, the budget, the problem, the time, auditing potential (for safety critical systems development), and persuasiveness.

56.3.2 Method as Achievement

It is over two decades since Suchman (1987)'s breakthrough critique of intelligent photocopiers. Suchman's research readily exposed how the plan recognition built into an intelligent photocopier was no match for the variety of real user behaviors. By treating a plan as a script, the intelligent photocopier was too rigid and unimaginative in its interpretation of user behaviors. Suchman argued that plans should be regarded as *resources* that have some fit to potential real world situations, but to achieve fit such plans need to be modified and extended to cope with the situated realities of human behavior. If we accept that it is highly unlikely that scripted plans in intelligent appliances can ever support management of situated human behavior, designers of usability methods face even bigger challenges, since usability work is far more challenging and complex than, for example, photocopying. We need to remove such inconsistencies within HCI thinking. If we cannot script user interaction, then we cannot script usability work. This has been shown for the user of user testing methods by Nørgaard and Hornbæk (2006), who studied how think aloud tests were conducted in seven Danish companies; they found that immediate analysis of observations made in think-aloud sessions was only performed sporadically, if at all. Usability approaches must then be designed, understood, and assessed within a framework that recognizes the situated nature of all human activities. Within such a framework, evaluation methods can never be anything more than weak prescriptions.

UIMs thus cannot be provided in any complete form by usability researchers or leading practitioners. Instead, what is provided are *approaches* that commoditize a set of resources by packaging them together under some method "brand name" (e.g., HE, CW). To return to a culinary analogy, what are called UIMs have much more in common with a Chicken Fajita *kit* on a supermarket shelf than with a Chicken Fajita *meal* in a restaurant. Consumers of the former expect it to be incomplete, and they expect to have to source some key ingredients and then prepare them, combine them with the kit, and cook them. Purchasers of the latter just expect to eat Chicken Fajitas and would be very dissatisfied if they had to prepare or cook anything. Evaluators who expect methods to be like completed meals will be similarly dissatisfied. They should expect *approaches* that require them to augment and adapt. With this in mind, we now further review UIM "ingredients," both those provided by "branded methods" and those developed within HCI research that can readily be combined within a range of evaluation resources.

56.4 RESOURCES FOR USABILITY INSPECTION

Attending to the resources present in (or absent from) descriptions of UIMs offers several benefits to researchers and practitioners. First, the important choices in usability inspection are among resources, not among methods. Second, resources may be configured and combined as evaluators see fit, which is overlooked when the focus is on methods. Third, some approaches to inspection require resources that are implicit or unusable, which a focus on resources can better expose. Fourth, focusing on resources allows sharper and more controllable research studies with a more realistic chance of benefitting practitioners. The studies can focus on specific aspects of usability work (resources) rather than incomplete combinations of resources (approaches).

56.4.1 Resources within Three Example Usability Inspection Methods

To exemplify the notion of resource in UIMs, we return to the three example UIMs outlined earlier in this chapter. First, for HE, we have already identified heuristics as the key resource. The two original papers on HE (Molich and Nielsen 1990; Nielsen and Molich 1990) listed nine heuristics (e.g., "simple and natural dialogue" and "prevent errors"). Subsequently, the heuristics have been through several iterations. Nielsen (1994b) did a post hoc analysis of 249 usability problems to find seven heuristics that could have predicted many of those problems and added three others to produce a more comprehensive set.

As noted earlier, HE also provides a procedural resource, but descriptions of it vary in their recommendation of the use of *task resources*. Originally, Nielsen and Molich (1990) mention no need for formal or informal understanding of tasks for evaluators. Later, Nielsen (1993) mentions that in contexts where evaluators lack *domain knowledge* (which Nielsen had shown to be a critical evaluation resource), they may be supplied with "a typical usage scenario" (p. 159) even though "evaluators are not *using* the system as such (to perform a real task)" (p. 159). Despite many studies of HE, we are unaware of any that have tried to manipulate task specification and analysis resources, even though Sears (1997) showed that combining CW's strict procedure with HE's loose sweep improves over the performance of HE alone.

In themselves, heuristics are not very usable. Both studies of, and practical introductions to, HE provide further tutorial material. Nielsen and Molich (1990) suggested that the heuristics could be presented to evaluators in "a single lecture" (p. 250). Nielsen (1994b) walked through a set of heuristics, explaining and exemplifying them almost 50 pages. Because of this variety in introductory material to HE, some experiments have used lectures and tailored material to standardize evaluator education (e.g., Cockton and Woolrych 2001, which identifies when task resources are essential to identify heuristic breaches). The extent to which such additional *knowledge* resources influence performance with HE is presently unclear.

Given the limited and varied resources associated with HE, it is no surprise that HE's resources alone cannot explain documented evaluation outcomes. Jeffries et al. (1991) asked evaluators note how they identified problems when using HE. Analysis of their notes showed that roughly a quarter of problems were found via "side effects" (e.g., from prior experience) and about a third of the problems were found "from prior experience with the system." Cockton and Woolrych (2001) demonstrated that many outcomes of HE could not be attributed to its resources, since evaluators successfully predicted usability problems with which no heuristic could be credibly associated, and thus had to inappropriately associate a standard heuristic with many of the one third of usability problem predictions that were confirmed by carefully focused user testing.

Turning to CW, as noted, its key resource is its prescribed procedure, based on a set of tasks, which answers four questions for each step of a task. In contrast to HE, CW's procedure is clear and always task-centered, although the final step of forming success/failure cases is not well described. CW forces use of task resources by requiring analysts to answer four questions for each task step, and then construct a success and failure case for the task. However, there is no support from CW for forming success and failure cases, nor is there support for task selection and specification. Even so, CW's success and failure cases are rare examples of an *analysis resource* for *problem elimination*, letting analysts argue that some apparent design flaws would not automatically cause severe usability problems. In contrast, most UIMs provide *discovery resources* that guide evaluators to find *possible* problems. To avoid a high proportion of *false negatives*, UIMs need to also provide *analysis resources* that support evaluators in deciding whether a found problem is *probable*, and thus should be reported, or *improbable*, and thus should be discarded.

The cost-benefit balance arising from CW's task-based procedures may be unfavorable. Early studies such as John and Packer (1995) found that going through CW was "very tedious." Early versions of CW required analysts to explicitly consider users' goals (e.g., Lewis et al. 1990; Polson et al. 1992). For example, the version presented by Polson et al. (1992) required three pages of questions for each action; one page to address

users' goals, one to cover choosing and executing the correct action, and a final page of questions covering the effect of taking the action on the user's goal structure. The overhead of CW and difficulty applying it led to the cognitive "Jogthrough" (Rowley and Rhoades 1992) and multimedia tool support for CW procedures (Rieman et al. 1991), which increased evaluation efficiency markedly, that is, they improved CW's cost-benefit ratio by adding a further *tool resource*.

A simpler version of CW was developed by Wharton et al. (1994), which de-emphasized "the explicit consideration of the user's goal structure," resulting in the current set of four questions, although a subsequent variation (Spencer 2000) reduced these down to two! Overall, later versions of CW reduced extensive preparation materials for analysts, making them less unwieldy as discovery resources, but perhaps losing some benefits that arise with interaction-centered methods, with some benefits preserved via the use of task descriptions, but others lost due to the progressive simplification of CW and the resulting loss of explicit resources for evaluation support.

Early versions of CW emphasized *theory as a resource*. Answers to questions should be supported by empirical data, experience, or scientific evidence, all knowledge resources external to CW, except for CE+, a theory of learning (Polson and Lewis 1990) that provided CW's foundations. For example, CE+ predicts that users new to a system will choose an action with a good match to their current goal. This reflects the relation between interaction-centeredness and grounding in theory, in that the system-centered methods such as HE are generally atheoretical. Later versions of CW backgrounded the CE+ theory of learning (Wharton et al. 1994; Spencer 2000), suggesting that there is a tendency to favor concrete (notations, questions) over abstract resources (theories, concepts) and practical ones (e.g., procedures) over propositional resources (e.g., knowledge, information).

As noted for the third example UIM above, MOT's key resource is the five metaphors and their associated key questions (Table 56.1). Metaphor choice was guided by considerations of how this resource would work for analysts. MOT's inventors argued that "use of metaphors as a communication device supports intuition and requires active interpretation; an effort orthogonal to developing inspection techniques that are more strictly formal and piecemeal analytical" (Frøkjær and Hornbæk 2008, p. 3). The extent to which the metaphors and key questions actually work in this way for analysts is, however, only investigated through summary measures of evaluation performance and through analysts' comments on MOT.

The procedure of MOT is similar to that of HE (Frøkjær and Hornbæk 2008), but contains more detailed instructions on how to conduct an evaluation (Hornbæk and Frøkjær (2002). The instructions mention that the analyst must use "typical tasks" to drive the evaluation, but does not provide any resources for identifying these. MOT thus provides similar task resources to CW, that is, stating that they must be used, but providing no support for selecting and specifying them.

In the above analysis, we have begun to distinguish different classes of resources. Two broad classes are *discovery*

TABLE 56.1
Summary of the MOT-Technique: The Five Metaphors, Their Implications for User Interfaces, and Examples of Questions to Be Raised

Metaphor of Human Thinking	Implications for User Interfaces	Key Questions/Examples
Habit formation is like a landscape eroded by water.	Support of existing habits and, when necessary, development of new ones.	Are existing habits supported? Can effective new habits be developed? Is the interface predictable?
Thinking as a stream of thought.	Users' thinking should be supported by recognizability, stability and continuity.	Does the system make visible and easily accessible the important task objects and actions? Does the user interface make the system transparent or is attention drawn to non-task related information? Does the system help users to resume interrupted tasks? Is the appearance and content of the system similar to the situation when it was last used?
Awareness as a jumping octopus.	Support users' associations with effective means of focusing within a stable context.	Do users associate interface elements with the actions and objects they represent? Can words in the interface be expected to create useful associations for the user? Does the graphical layout and organization help the user to group tasks?
Utterances as splashes over water.	Support changing and incomplete utterances.	Are alternative ways of expressing the same information available? Are system interpretations of user in-put made clear? Does the system make a wider interpretation of user input than the user intends or is aware of?
Knowing as a site of buildings.	Users should not have to rely on complete or accurate knowledge—design for incomplete-ness.	Can the system be used without knowing every detail of it? Do more complex tasks build on the knowledge users have acquired from simpler tasks? Is feedback given to ensure correct interpretations?

and *analysis* resources, which respectively support finding possible usability problems and deciding on their probability and severity. *Task specification resources* are a subclass of discovery resources when used to guide problem discovery. However, information on *task frequency and criticality* can be regarded as analysis resources when they are used to determine the severity of a probable problem. Here, they would also be instances of *severity rating* resources that *rank* probable problems by their *likely impacts* on users. Other examples of evaluation resource include *matching and merging* resources (to *consolidate* the predictions of one or more evaluators into a single set of probable problems) and *expressive resources for reporting* (to support merging/matching and/or communication of the final problem set). Further types of evaluation will be identified in the next two subsections.

56.4.2 Resources within Other Usability Inspection Methods

We now introduce four further UIMs. The first, *Heuristic Walkthrough* (HW: Sears 1997), is a hybrid approach that combines the resources of HE and CW. HW also requires a prioritized list of user tasks, which should include frequent or critical ones, and may also include tasks designed purely to ensure coverage of the system being evaluated. HW thus provides more guidance on task selection than HE, CW, or MOT. This *discovery* resource is complemented by a further procedural resource that provides additional support for discovering possible problems by combining and extending the basic discovery procedures of CW and HE. It has two phases: task-based and free-form. In the first phase, evaluators explore tasks using a set of thought provoking questions derived from CW. They are free to explore the tasks in any order, spending as long as they need, but they should be guided by task priorities. In the second phase, evaluators are free to explore the system. They use the set of thought provoking questions *again*, plus HE's heuristics. Sears (1997) compared HW against CW and HE. HW found more actual problems than CW and had fewer false positives than HE. Overall, Sears' merge of HE and CW improved evaluator preparation and discovery resources, with evidence of improved analysis resources (reduction in false positives), which may be due to the initial CW inhibiting discovery of unlikely problems. HW appears to encourage more self-reflection on the part of evaluators relative to HE, but this is not due to the "method as a whole," but to a single procedural resource (two phase problem discovery structure).

Turning to our next UIM, *Ergonomic Criteria* (EC: Bastien and Scapin 1995; Scapin and Bastien 1997) uses a set of 18 ergonomic criteria. They were formed by reviewing around 800 existing guidelines and experimental results (Scapin 1990). Not surprisingly, HE's heuristics and EC are often very similar, for example, Immediate feedback (HE: visibility of system feedback) and Quality of error messages (HE: help users recognize, diagnose, and recover from errors). Each has a definition, a rationale, and examples of guidelines plus comments that include help to disambiguate any related criteria. There are eight main (top level) criteria: guidance, user workload, user explicit control, adaptability, error management, consistency, significance of codes, and compatibility. Five of these are further subdivided, resulting in 13 second-level criteria: prompting, grouping and distinguishing items, immediate feedback, clarity, brevity, mental load, explicit actions, user control, flexibility, users' experience management, protection from errors, error messages, error correction. Further subdivision results in overall 18 criteria at three levels of abstraction and structure.

While some accounts of HE provide similar support (e.g., Nielsen 1993), EC's consistent structures are rare for UIMs. EC's structure is likely to assist evaluator preparation for using it. EC thus considers evaluators' learning needs and provides structure and detail to prepare them for using it. Important preparation resources including scoping, axiological and knowledge resources (Woolrych, Hornbæk, Frøkjær, and Cockton 2012), which respectively indicate the scope of a UIM, the values underlying its design, and the knowledge required to fully understand it. Note that these resources have an impact before inspections are attempted in better *preparing* evaluators. Preparation resources are distinct from discovery or analysis resources, although both of the latter clearly interact with the former. EC also requires evaluators to be provided with description of the *purpose* of the product, a preparation resource that is lacking in HE, CW, MOT, and HW. EC also provides procedural resources that support discovery and analysis: a first phase follows a task-based walkthrough, while a second phase focuses on problem diagnosis.

Our reviewed UIMs so far have distinct motivations. HE focused on discounting evaluation costs, CW on learning walk-up-and-use systems, MOT on empathy with user behaviors, EC on supporting evaluator learning, and HW on trading off one evaluation cost (false negatives) against another (first pass walkthrough costs). Our next reviewed UIM, *Cognitive Dimensions* (CD: Green and Petre 1996), sought to improve evaluators' conceptual vocabulary, moving them away from technical surface features (Green 1991). CD attempts to capture a set of orthogonal dimensions for restricted set of systems called "notations," which are used to design information structures, for example, programming languages and spreadsheets. These dimensions are not necessarily criteria for a design but a characterization of the design space.

Sample of dimensions are shown in Figure 56.3. The dimensions embody a theory about how people use notations

Hidden dependencies occur when one cannot see how a value is calculated, for example, a spreadsheet where a small change can have unexpected effects. All dependencies that are important to the user must be accessible.

Premature commitment occurs when a user is forced to make a decision before the necessary information is available.

Viscosity is resistance to change or a measure of how much work is needed to achieve a change.

Abstraction gradient: An abstraction is the grouping of objects into one object, which can make a system easier to understand.

FIGURE 56.3 Example cognitive dimensions.

to build information structures. A simplified theory of action would involve people translating goals into specifications, then into actions, and executing them. Rather, people work bottom up, top down—goals and subgoals may be attacked at any moment. Green (1991) claimed the preferred strategy is opportunistic where high- and low-level decisions are mingled; commitment can be weak or strong and development in one area may be postponed because of problems foreseen in other areas. The vocabulary of dimensions may act as a discovery resource, sensitizing analysts to generic interaction difficulties. However, once associated usability problems are discovered, there appears to be little further support for problem confirmation or elimination. Even so, as with CW's CE+, CD provides *theoretical knowledge resources* that provide valuable *preparation* resources for evaluation, without which CD would provide little guidance on applying the dimensions. A notation called "ERMIA" (Green 1991) was proposed as a method of exploring dimensions. Interestingly, this notation seems to have been developed (Green and Benyon 1995, 1996) into a stand-alone UIM without dimensions. ERMIA is our second example of a tool resource (Rieman et al's automated CW was the first). Thus, as well as practical knowledge resources (list of cognitive dimensions), CD provides a tool resource that supports evaluation practices. The theoretical focus on opportunistic planning draws on work by Green's colleague Young at MRC Cambridge (Young and Simon 1987), and thus strongly resonates with the position on usability work in this chapter that evaluators do not systematically follow methods top down.

Our final reviewed UIM, *Pluralistic Walkthrough* (PW: Bias 1994), originated at IBM. In PW, an administrator leads a team of three diverse participants (a representative user, a product developer, and a usability specialist) who pretend to be system users. Each is given a set of hard copies of screens and task descriptions. For each screen, the participant is asked to write down in as much detail the next action (or set of actions) that the user would take to achieve the task and any comments they have about the screen. Once each participant has written down the next action, the screen is discussed. When the representative users have finished talking (e.g., about problems), the usability specialists and product developers talk (e.g., to explain the rationale of certain features). No assessment of this UIM has been published; however, there are clear potential benefits from *preparation* resources (hard copies of screens, task descriptions), and the combination of representative users, product developers, and usability specialists provides further discovery and analysis resources in the form of *knowledge of users* (from representatives), of the *product* being evaluated (from developers, who also provide *technical* background knowledge), and of *tasks* and *interaction* (from usability specialists). PW is distinct in adopting a *social* strategy to provide evaluation resources. Usability specialists draw up task descriptions, developers provide copies of relevant screens for the task, representatives provide user knowledge, and the administrator provides procedural knowledge of PW. In some ways, PW anticipated the recent and long overdue "turn to the social" in HCI evaluation research and practice, motivated by the opportunity to draw on evaluation resources that already exist within the project context. Unlike Spencer's (2000) restrictions on CW, PW treats project social interactions positively, rather than a potential negative source of disagreements.

56.4.3 Generic Resources for Usability Inspection

In addition to resources that are closely associated with particular UIMs, many resources are discussed outside the context of particular UIMs or are common to several. Table 56.2 summarizes some of the most important generic resources for usability inspection. First, *task selection and specification* are often key to preparation and provide a resource for many inspection and model-based approaches. As with individual differences, the selection of particular tasks has a large effect on the interaction with computers and hence on evaluation outcomes (Lindgaard and Chattratichart 2007). Sears and Hess (1999) described how the amount of detail in task description affected which problems were found in CW. Analysts who used detailed task descriptions found significantly more

TABLE 56.2
Generic Resources for Usability Inspection

Resource	Purposes	Examples
Task selection	Selecting and specifying tasks for inspection or user testing	Sears and Hess (1999)
Task walkthrough	Supporting inspection methods with ways of going through tasks and interfaces	Sears (1997)
Heuristics	Resources for discovering and thinking about usability defects	Hvannberg, Law, and Lárusdóttir (2007); Mankoff et al. (2003); Somervell and McCrickard (2005)
Reporting formats	Helping communicating problems and solutions for subsequent analysis, evaluation auditing, iteration, and customer communication	Cockton, Woolrych, and Hindmarch (2004); Capra and Smith-Jackson (2005); Theofanos and Quesenbery (2005)
Problem analysis and classification	Ways of classifying problems so as to analyze, merge, or reject them	Andre et al. (2001); Cuomo and Bowen (1992)
Problem merging	Identifying similar problems, for instance, through tool support or using different definitions of similarity	Connell and Hammond (1999); Hornbæk and Frøkjær (2008a); Howarth, Andre, and Hartson (2007)

low-severity problems, but fewer problems relating to feedback from the system, compared to evaluators who used less detailed descriptions. Similarly, obvious differences have been shown to exist in user testing between user-generated tasks and set tasks, and also between open-ended and closed tasks (Spool and Shroeder 2001), but little research has been conducted on the impact of task types within usability inspection.

Similarly, CW and related UIMs are the only ones to provide resources for guiding evaluative task walkthroughs. There is also considerable scope for research here to compare the costs and benefits of different task walkthrough procedures. For example, when using CW, once one of the four questions has been answered in the negative for one step, it may be the case that all subsequent task steps should have negative answers for the same question. Alternatively, the answer to a question may have to be deferred for several steps, until it can be confidently answered. This is especially the case for the fourth question: *if the correct action is performed, will the user see that progress is being made toward solution of the task?* This may not become clear for several steps after the correction action is performed. There is thus scope for streamlining task walkthroughs by propagating/deferring answers to later steps. However, the effectiveness of such streamlining needs to be demonstrated through research. The only relevant research here currently relates to the presence or absence of task walkthroughs (e.g., HE vs. HW, Sears 1997).

While HE provides the best known set of heuristics for usability inspection, *alternative heuristics* are possible: Hvannberg, Law, and Lárusdóttir (2007) have compared Nielsen's heuristics to Gerhardt-Powals' cognitive engineering principles. Mankoff et al. (2003) developed and tested a specialized set of heuristics for ambient displays. Somervell and McCrickard (2005) have developed similar heuristics for large-screen information exhibits through the use of a structured process that relies upon critical parameters. Muller et al. (1995) rightly claimed that Nielsen's (1994b) heuristics ignore usage contexts. They added three extra heuristics to assess how well a design fits user needs and the work environment:

1. Respect the user and his or her skills
2. Pleasurable experience with the system
3. Support quality work

These studies have shown how specific heuristic resources impact evaluation performance. Heuristic sets vary considerably in their generality and derivation (e.g., empirical for HE and EC, systematic focus for Somervell and McCrickard, theory for CD and Gerhardt-Powals), as well as in their structure and presentation (e.g., HE vs. EC). There is again considerable scope for practically oriented research that can isolate the varying impact of overall philosophies, systematic foci, structure and presentation, and formal derivation of heuristic sets. Effective innovation in any of these areas would improve evaluation performance using heuristic sets.

Task selection and task walkthrough procedures, plus heuristic focus, derivation and presentation are relevant to only UIMs that include or require them as resources. However, some resources are rarely provided or required by UIMs, and yet are unavoidable. For example, the way that usability problems are reported plays an important role in usability inspection, and thus report formats are potentially a key evaluation resource.

Cockton et al. (2003) demonstrated that an *extended problem reporting format* can positively impact evaluator performance on false positives and appropriate HE use, with results here replicated in Cockton, Woolrych, and Hindmarch (2004). This extended format required evaluators to explain how they found a possible problem and how they decided whether to confirm it as a probable problem or discard it as an improbable one. This appeared to have improved evaluators' self-awareness and repertoires when *looking for* possible problems, making problem discovery less tacit. It also appears to have improved self-awareness and repertoires when *analyzing* possible problems, encouraging them to carefully consider whether a possible problem should be treated as probable or improbable. More explicit behaviors appear to have been prompted by the need to explain problem discovery and analysis, confirming the wisdom of an earlier conjecture that having justifications for why something is a problem is important (Jeffries 1994); the downstream utility of problem descriptions seem to increase with such justifications (Hornbæk and Frøkjær 2006).

Structure and its content are generally created simultaneously for problem report formats, but adding a single content item to an existing format can significantly improve evaluation quality. For example, evaluators in a think aloud test were instructed to consider *business goals* while evaluating and to report, together with typical parts of problem reports, the importance of a problem in relation to business goals (Hornbæk and Frøkjær 2008b). Compared to a control group, this format made evaluators report fewer problems, but the problems reported were seen as being of more utility in the development process. Although the benefits of this report item were demonstrated for user testing, it is reasonable to assume that similar benefits would arise if used alongside UIMs and might be particularly useful for evaluators with limited knowledge of the business goals for an application. Such a small change with a potential large impact further illustrates the value of focusing UIM research on the impact of specific resources, rather than on the overall impact of highly variable incomplete "methods."

A further specific report format element is the *type* of the usability problem, for example, its severity class (e.g., Jeffries et al. 1991). Types may be based on a range of constructs, for example, by relation to stages in Norman's (1986) theory of action. Using such a coding scheme, a study (Cuomo and Bowen 1992, 1994) found that an early version of CW tended to focus on problems related to the formation of actions and was poor at finding problems concerned with evaluating the display. A similar coding approach was used for an assessment of a later version of CW (Sears and Hess 1999), which demonstrated the impact of short and (very) detailed task descriptions on analyst performance. The impact of each resource was assessed by coding problems by the CW question relevant to the prediction. Short task descriptions resulted in relatively more predictions related to finding actions (CW Question 2). Detailed descriptions (to the level of naming and

locating controls) resulted in relatively more feedback-related predictions (CW Question 4). Sears and Hess (1999) explain the differences through the impact on how evaluators discovered problems. Detailed descriptions led to finding fewer action specification problems than short descriptions did, but left evaluators with the energy to look for feedback problems.

In most usability work, individual problems are predicted or discovered that are instances of more general and/or frequent user difficulties. Such individual problem instances need to be matched to reduce a large set of individual predictions or user difficulties to a master set. *Problem matching and merging* is a major source of confounds when comparing usability methods (Cockton and Lavery 1999; Connell and Hammond 1999; Hornbæk and Frøkjær 2008a; Lavery, Cockton, and Atkinson 1997). Hornbæk and Frøkjær (2008a) showed how differences in instructions for how to match usability problems significantly affected the number of problems that could not be matched with other problems (in a set of 55 problems): The degree to which a single problem type emerges from UIM use matters a lot to research, but in this study, it was strongly affected by how matching was performed. This is not only a problem for research, since usability practitioners must match and merge problems for inspections with multiple evaluators, suggesting that matching and merging are unavoidable in usability work. Once multiple evaluators are used as recommended for HE (Nielsen 1994), the evaluators require analytical resources for matching and merging problems. These analysis resources may be procedural (e.g., for use by a chief evaluator), structural (based on matching report format elements), or social (e.g., merging by a group of evaluators in face-to-face discussions). Matching and merging resources determine the final problem set (types with instances) that forms the basis for understanding and addressing usability problems within iterative development. They are thus no less vital a resource than heuristics (criteria, dimensions), task selection (as in HW), or task walkthrough procedures (CW, HW, PW), and yet, along with problem reporting formats, they have received very limited attention from UIM developers and assessors.

56.4.4 Evaluation Resource Types for Usability Inspection

We can thus see that a wide range of resources will impact evaluators, with some resources being particularly relevant to specific steps in usability inspection, with which we can associate a *logical* (rather than temporal) structure with logically distinct phases that overlap during inspection. Inspection is essentially a search problem. Evaluator behavior is close to the generate and test strategy from early artificial intelligence research, where possibilities are first generated (e.g., moves in a chess game) and then tested to identify possibilities with specific attributes (e.g., the best next move). Note that the role of evaluation does not extend to fixing problems, even though methods such as *Rapid Iterative Testing and Evaluation* (RITE) (Medlock et al. 2005) have problem fixing within scope. RITE is structured around three questions: Is it a problem? Do we understand it?, and Can we fix it? Only the first question and some of the second are within scope of evaluation methods. The rest of RITE (really understanding and fixing problems) is an *iteration* method with (empirically) grounded redesign decisions, based on all relevant project information.

In the first *logical phase*, evaluators study UIMs and related resources. Also, each time a UIM is configured, augmented, and applied, evaluators must study the target design in the form provided, as well as the intended usage context. These *preparation* steps are followed by two logical generate and test phases of UIM application, which is followed by a final reporting phase. We can thus model usability inspection as having four distinct logical phases: (1) analyst preparation, (2) candidate problem discovery, (3) confirmation or elimination of candidate problems, and (4) problem reporting, with initial support for problem understanding and change recommendation. This results in four major *uses* of evaluation resources: preparation, discovery, analysis, and reporting. Evaluator preparation has been given limited attention in the UIM research literature. A notable exception is the preparation of tutorial material for HE (Lavery, Cockton and Atkinson 1996a), CD (Lavery, Cockton and Atkinson 1996b) and other UIMS. This HE tutorial was used in Cockton and Woolrych (2001).

Most UIMs focus on discovery and analysis. The DARe model (Discovery and Analysis Resources, initially DR-AR, Woolrych and Cockton 2002) was formed when exploring HE's scope and accuracy (Woolrych 2001), especially the causes of false positives. In any predictive method, analysts inspect a design for potential usability problems. Where multiple evaluators use a UIM, their problems must be merged into a "master" problem set. This will invariably contain false positives (problems predicted by evaluators that are not problems for actual users). Now, these can *only* be problems that have been *incorrectly analyzed*. They could not have been incorrectly discovered, as *the only discovery error is a genuine miss*—analysis is responsible for all other errors. Not all false positives are predicted by every evaluator, leading to a question: can some analysts correctly eliminate these through a process of analysis as true negatives? This conjecture arises directly from the DARe model, with its two logical phases, where analysts first discover possible usability problems, then through analysis either confirm them as probable problems or eliminate them as improbable problems.

The DARe model guided research to focus on the evaluation resources used in usability inspection and particularly those that were not provided by the UIM in use. Studies of problem reports have identified seven groups of knowledge resources that can influence problem discovery and analysis (Woolrych, Cockton, and Hindmarch 2005):

- **User** (knowledge of/beliefs about users, especially experience and abilities)
- **Task** (knowledge of what users want to do and how they prefer to accomplish this)
- **Domain** (domain knowledge that is specific to the system being evaluated)
- **Design** (knowledge and experience of interaction design principles or beliefs in lieu of)

- **Interaction** (knowledge of how humans actually interact with computers)
- **Technical** (knowledge of platform technologies such as browsers and toolkits)
- **Product** (information about the system and its capabilities)

These resources are known as *Distributed Cognitive Resources* (DCRs), so called because they are unevenly and unpredictably *distributed* across project contexts (UIMs, evaluators, other project resources). Evaluator effects (Hertzum and Jacobsen 2001) can be explained via different individuals possessing and/or accessing DCRs in different ways during inspection. All types of DCR have already been encountered above when reviewing UIMs, but this empirical study (Woolrych, Cockton, and Hindmarch 2005) showed all in use with HE, which provides few of them directly. In principle, MOT promotes *user* resources, as did early versions of CW, and PW supplies them directly. CW, HW, MOT, and PW require *task* resources. Nielsen's studies of HE usage have highlighted the importance of *domain* resources, as has Cockton and Woolrych (2001). HE and EC distill both *design* and *interaction* resources into heuristics and criteria, respectively. PW also provides both, as well as *product* and *technical* information, through its involvement of specialists. CW and CD embody theories of interaction. EC requires some product information to be provided. However, HE neither provides nor prompts for all seven groups of DCR, and yet Woolrych, Cockton, and Hindmarch (2005) were able to identify multiple instances of each in HE problem reports. Student evaluators were clearly augmenting HE with personal resources (they lacked a project context that could have provided any additional DCRs). Two evaluators in Cockton and Woolrych (2001) were asked to not use a specific UIM, but were instead given specific tasks to complete. Inspections that require no resources beyond the evaluator are called *expert inspections*. Here, expert inspections were supplemented with tasks based on domain goals (copying diagrams with a drawing tool), discovering problems that over 90 other student evaluators using HE did not.

DCRs can be used for both discovery and analysis. Extended structured problem reports that require evaluators to externalize their discovery and analysis steps are needed to distinguish the use of a DCR for discovery from use for analysis. System-centered resources (design, product, technical), as well as user-centered ones (user, task, domain, interaction), can guide discovery. The use of knowledge resources can be structured or unstructured, resulting in four broad classes of *problem discovery method*. There are system-centered unstructured and structured methods: *system scanning* and *system searching*. HE distills interaction design knowledge into heuristics. If HE in any way supports problem discovery, it is through heuristics that focus attention on design features such as error messages and navigation. Similarly, there are user-centered unstructured and structured discovery methods: *goal playing* and *procedure following*. These two discovery methods either use contextual information about users, tasks and domains or they substitute evaluators' beliefs about them. CW relies exclusively on task information to find problems. Appropriate task, technical, and domain knowledge can aid analysts in valid problem discovery and analysis. Knowledge of tasks allows analysts to discover problems that require sophisticated levels of interaction with the system before they become obvious. Simple system-centered discovery approaches would be unlikely to discover such problems.

Incorrect beliefs about users often lead to false positives, whereas product and interaction knowledge can avoid them (through correct elimination of improbable problems). Knowledge of how people interact with computers can imply that apparent design flaws will have no impact, for example, because users will never notice them (selective attention) or because they can learn what they need through interaction (distributed cognition). Technical and design knowledge can be used to correctly confirm a problem. However, reuse of resources in both discovery and analysis introduces risk of confirmation bias. Using the same resources in analysis as discovery adds nothing and is thus a source of false positives (Cockton, Woolrych, and Hindmarch 2004).

56.4.5 Project-Based Resources for Usability Inspection

Methods are only one aspect of usability work and thus successes or failures cannot wholly be explained in terms of resources within approaches. Instead, resources within the surrounding project context can have as much, if not more, influence on success or failure than the textbook resources of evaluation approaches in HCI.

We make no attempt here to survey all the factors that have been shown to shape usability work, but instead focus on some research that considered the broader context of method use. Theofanos and Quesenbery (2005), for instance, listed a variety of factors that influence how to report results, including the size of the recipient company, the kind of products evaluated, the audience, and the existence/absence of a formal usability process. Similarly, the software development approach used in a development project will influence usability work. Sy (2007) described how various approaches to usability evaluation had to be modified to fit the agile software development approach. For instance, reporting of problems was done through the daily scrum meetings and in process planning sessions, rather than through reports, and thus specific evaluation resources were not needed here (although usability roles may well have kept their own informal records).

We now quickly list several factors of which we are aware from research that has addressed how project- and organization-specific factors influence configuration and combination of evaluation resources. These include the following:

1. Client needs and expectations
2. Design purpose and vision
3. Project-, product-, and service-specific prioritization criteria
4. Business context
5. Budgetary and other logistical resources (e.g., time available)

6. Project leadership and design champions
7. Development approach and stage
8. Relation of usability work to the overall software development approach
9. Experience and competence of usability practitioners
10. Training and tutorial support on evaluation approaches
11. Professional/specialist education on general discovery and analysis resources in evaluation
12. Field research methods (users, tasks, etc.) and result communication formats
13. Task specification and notations used at design stage
14. Participant and evaluator recruitment strategies
15. Alignment of design purpose and evaluation purpose

The first five factors relate to the client and their economic/market context (or policy context for public sector projects). The next six factors (6–11) relate to the development team, with a specific focus on management and expertise. The last four factors (12–15) relate to design activities (last two focus on evaluation). These factors, and many others not listed above, severely limit the ecological validity of formal comparisons of evaluation methods, even in the unlikely event of confounds being so well managed that no issues could arise here.

Few factors above could be addressed substantially by generic HCI evaluation resources. All, however, are present to some degree in actual project and organizational contexts. Currently, the bulk of the resources that are critical to working out methods within usability work are external to documented HCI evaluation approaches. Given this, academic attempts to "compare methods" are bound to "fail the practitioner" (Wixon 2003), since as far as providing comprehensive support for usability work is concerned, all current documented approaches have already failed practitioners to some extent by not providing them with comprehensive resources. However, much of the success of "discount methods" such as HE may be due to the sparseness and flexibility of the provided resources, which let skilled evaluators fill gaps with resources from personal, project, or organizational sources.

56.5 ASSESSING AND COMPARING USABILITY INSPECTION METHODS

Having identified the broad range of evaluation resources that are needed, but with few provided directly by UIMs, it should be clear that assessing and comparing UIMs could be a hopeless endeavor, since what actually gets used is *never the UIM alone* (at least for current UIMs), since evaluators must draw on DCRs beyond a UIM to be able to carry out usability work. However, it is still worth reviewing the two main broad approaches for assessing and comparing UIMs—one qualitative and the other quantitative. *Qualitative* approaches code predictions as a basis for scoping the capabilities and defects of UIMs. *Quantitative* approaches relate the performance of one method to another, often expressed as a percentage, for example, to express proportions of correctly predicted problems. We first review quantitative approaches and then review qualitative ones. The critiques of both approaches to assessing UIMs as complete wholes adds to the insurmountable challenges for direct assessment and scoping of "pure" UIMs, which are next summarized. With this critique in place, we review how resources beyond the assessed and compared UIMs can account for differences in evaluator performance as readily, if not more so, than the few evaluation resources that a UIM actually provides.

56.5.1 Summative Assessment and Comparison

The fundamental purpose of usability inspection is to find and report any potential user difficulties for remedy in redevelopment. Ideally, a UIM would find *all* usability problems that may arise from use of an interactive system; in other words, it would be *thorough* and, hence, not miss any elements that may cause user difficulties. The final problem set achieved through usability inspection would also only comprise *valid* problem predictions, with no false alarms reported. Consequently, finding all problems without false alarms, the result of a usability inspection would be wholly effective.

Unfortunately, any predictive methods such UIMs are not completely thorough nor are the predictions always valid with the subsequent impact on UIM effectiveness. *Validity*, *thoroughness*, and *effectiveness* are quantitative measures of UIM capability for which Sears (1997) suggested the following formulae:

$$\text{Validity} = \frac{\text{number of } \textit{real} \text{ usability problems } \textit{found} \text{ by UIM}}{\text{number of usability problems } \textit{predicted} \text{ by UIM}}$$

$$\text{Thoroughness} = \frac{\text{number of } \textit{real} \text{ usability problems } \textit{found} \text{ by UIM}}{\text{number of } \textit{real} \text{ problems that } \textit{exist}}$$

$$\text{Effectiveness} = \text{Validity} \times \text{Thoroughness}$$

The scores derived from the formula are between 0 and 1, where 1 would be the *perfect* score. Such scores are often expressed as percentages (e.g., 0.5 = 50%). For validity, a perfect score (of 1) would mean *all* problems found by the UIM would be real problems, that is, with no false alarms. Similarly, if the count of real problems found by the UIM equals the count of problems that exist, this too would result in a perfect thoroughness score of 1. With perfect thoroughness and validity, effectiveness too would also be a perfect 1 ($1 \times 1 = 1$), that is, *all predictions valid, no false alarms, all problems found*. With thoroughness of 0.7 and validity of 0.5, effectiveness would be 0.35.

Many assessments of UIMs have only used thoroughness as a quantitative measure and have shown, for example, how the use of multiple evaluators increases thoroughness (Nielsen and Landauer 1993). However, when validity is also calculated, the benefits of rising thoroughness are soon undermined by falling validity and hence reduced effectiveness (Cockton and Woolrych 2002). Giving equal weight to thoroughness and validity may not be suitable for all

evaluation contexts (Hartson, Andre, and Williges 2001). However, relying solely on thoroughness and validity measures is inadvisable: as well as it being extremely difficult to calculate validity, they have no diagnostic power for explaining a lack of thoroughness or validity, and thus cannot guide the selection of complementary methods that can compensate for a UIM's weaknesses.

Two values in Sears' equations present severe research challenges. First, the denominator for thoroughness (number of real problems that exist) relies on the implicit *closure* of "all of the usability problems for a design," but "all" can never be determined confidently: there can always be undiscovered problems. All thoroughness measures are always upper bounds that reduce as newly discovered real problems increase the denominator. For the validity numerator (number of real usability problems found by UIM), we only need to find real usability problems that match those predicted by a UIM. Hence all validity measures are always lower bounds and increase if any newly discovered real problems match predictions, increasing the numerator.

While Sears' formulae are attractive and offer the simplest possible way of comparing UIMs, they rely on closures that mostly exist in mathematics (closure of a set under some operation). In the real world, we use the word *all* for *enumerations*, such as "prohibited items in carry-on baggage," where closure results from human decisions, and even this uses very broad categories (e.g., ammunition; automatic weapons; axes and hatchets, unless part of aircraft equipment; billiard cues; billy clubs; and blackjacks). However, *all* in the context of thoroughness has the sense of a closure and not an arbitrary human enumeration. Such conceptual dead ends are compounded by miscounts of problems that are discovered through user testing (the default source of number of *real* problems that *exist*). Evaluators appear to miss most problems in video data (Jacobsen, Hertzum, and John 1998), and further *miscounts* may arise from poor merging of usability problem sets (Cockton and Lavery 1999). Connell and Hammond (1999) showed that the impact of multiple evaluators on cumulative problem counts depends on how predicted problems are merged, resulting in a slower increase in cumulative thoroughness than showed in Nielsen and Landauer (1993). Also, specificity of description determines problem count according to Cockton and Lavery (1999). As usability problems become more specific, problem counts rise. For example, "misleading status bar messages" is one problem, but "incomplete prompts" and "technical vocabulary in message" are two, and more would result if we distinguished between prompt problems such as "no instruction to close free form with double click" or "no instruction to use control-click to delete vertices" (for drawing objects).

For thoroughness, *asymptotic testing* has been proposed as a means to ensure the highest possible denominator (Hartson, Andre, and Williges 2001). However, this is generally understood as continuing to test additional users using the same test protocol until no new problems emerge. Changing the test protocol, such as between different sets of fixed tasks, between free and fixed tasks, or between individual think aloud and peer tutoring, also reveals new problems (Lindgaard and Chattratichart 2007). Also, adding analysts and/or applying a structured method such as SUPEX could extract more problems from video data. As with most attempts at closure, methods to find *all* problems simply expose a further unachievable closure, that is, in *all of the ways to find all* the usability problems. However, maximizing the numerator for *validity* is more tractable. When real problems are found by fixed task testing, a *falsification methodology* can expose users repeatedly to predicted problems (Woolrych, Cockton, and Hindmarch 2004). Missed problems here will have two types of cause: errors in test task planning (which can be fixed) and an inability to put the system in a state where the problem would appear (which can be very difficult to achieve; Lavery and Cockton 1996).

Some quantitative measures are not prone to insurmountable mixes of conceptual and practical impossibilities. Cockton and Woolrych (2001) coded heuristic applications for nonbogus predictions as being appropriate or inappropriate, based on explicit criteria from the HE training manual (Lavery, Cockton, and Atkinson 1996a) provided for analysts. This provided conformance questions that stated, "What the system should do, or users should be able to do, to satisfy the heuristic" (p. 4). Such questions are answered with conformance evidence, such as the "design features or lack of design features that indicate partial satisfaction or breaches of the heuristic" (p. 4). For many heuristic applications, these criteria were clearly ignored. Only 39% of the heuristics applied to nonbogus predictions were appropriate (and only 31% for successful predictions). Such measures of appropriateness are important in identifying the extent to which problem discovery and analysis are due to UIM or other evaluation resources. Heuristics tended to be best applied to predictions that turned out to be of low frequency and/or severity. These were likely to be predictions that seemed less probable than ones that turned out to be of high frequency and/or severity, which presumably were so obvious that the 13 heuristics (Nielsen 1994a, plus three specific to visualization; Lavery, Cockton, and Atkinson 1996a) were not properly reviewed to strengthen analysis. Furthermore, given HE's known limited coverage, there may be no appropriate heuristic for some predictions. Nielsen (1994b) found that seven factors could only account for 30% of the variability in his corpus of 249 usability problems, and 53 factors were needed to account for 90%, too many for HE. The names of the seven factors, which were chosen by Nielsen, included "visibility of system status," "match between system and real world," and "error prevention." This supported his claim that the factors (the first seven heuristics in Figure. 56.1 along with 8 and 9) form a basis for a set of heuristics that has remained the main resource for HE; the 10th, "Help and Documentation," was later added. It is not clear, however, how much the additional three heuristics extend the coverage of the sample problem set beyond 30%, but this is very unlikely to reach 40%. As a result, HE cannot cover most likely usability problems, and the ones it does cover are not always high frequency ones.

56.5.2 Qualitative Approaches to Coding Usability Problems

Quantitative measures can be seductive, so it is important to approach them critically. Even if simple scores for thoroughness or validity could be reliably calculated (and to be clear, they cannot), identical scores could hide major differences in UIM performance (Woolrych and Cockton 2000). For example, two UIMs could have identical thoroughness scores, but one could systematically miss most severe problems, which has been shown to be true for HE (Cockton and Woolrych 2001) by *coding* successful predictions and missed problems for *discoverability*. The easiest problems to discover are *perceivable*: these can be seen at a glance. Next comes *actionable* problems: these can be discovered after a few simple actions (e.g., a mouse click). Hardest to find are *constructable* problems, which require complex task scenarios to reveal them. 80% of problems missed by evaluators using HE were constructable, as opposed to 7% of successful predictions. HE's limited use of task knowledge resources is responsible for missing usability problems that only arise in interaction sequences longer than a few simple actions.

Discoverability is a *qualitative* measure, requiring UIM assessors to code a problem as perceivable, actionable, or constructable. Useful codes delve below the surface of superficial quantitative measures to reveal the impact of a UIM's (lack of) evaluation resources. However, the ability to code for a specific measure depends on the contents of the reporting formats for usability problems. In an experiment to establish common ground for method assessment, Cockton and Woolrych (2009) analyzed eight multilingual usability report problem sets from various evaluation methods (HE, EC, user testing, expert inspection) in a variety of formats (e.g., research problem set, consultancy report). The problem reports were inspected for common elements that could be a basis for comparison through problem coding, including the following:

- Description of the usability problem
- The context in which the problem was discovered
- Evidence of how the usability problem was discovered
- Actual or predicted user difficulties relevant to the problem
- Evidence of the frequency of the problem
- Any suggested solutions to the problem

The only common element in all of the problem report sets was a problem description! The context of problem discovery was quite common in most (but not all) problem sets, but often within a problem description. Thus comparing UIMs on the basis of qualitative measures requires appropriate problem reporting formats. Identification of DCRs in research by the first two authors, and analysis of interactions between them, depended on the ability to code problem reports for the use of a range of evaluation resources and the ability to distinguish their separate use on discovery and analysis, which was not always possible. Severity codings were used in early studies of HE by the first two authors to reveal the interaction between appropriateness of heuristic use of the impact of actual problems. This suggested that appropriate heuristic use was a form of confirmation bias arising when considering predictions that turned out to have low frequency and/or severity in the set of actual problems revealed through falsification testing (Cockton and Woolrych 2001).

56.5.3 Why Comparisons of Usability Inspection Methods Fail to Produce Reliable Results

Initially, problems in comparing UIMs were attributed to poor experimental design. Gray and Salzman (1998) reviewed the validity of five major studies in the literature that compared the performance of different evaluation methods. They based their review on Cook and Campbell's (1979) four main types of validity: statistical conclusion validity, internal validity, construct validity, and external validity. They added a fifth type: conclusion validity. Gray and Salzman found validity problems with each study reviewed. For example, Jeffries et al. (1991) compared CW, expert inspection (but called HE), guidelines, and user testing and found expert inspection to be superior. However, the study used few evaluators and thus it is not clear if the effects of the study (e.g., the superiority of expert review) occurred by chance and lacked *statistical conclusion validity.*

Internal validity is threatened when the effect supposedly caused by a variable under manipulation is actually caused by a confounding variable. One study where expert review was shown to be superior may, however, suffer from internal validity because the experts were given a 2-week period at their own pace to complete the evaluation, whereas evaluators in the other conditions had far less time. The worst performing methods were used by software engineers with less usability expertise than the HE group. Here, *time* is revealed as an evaluation resource as well as additional DCRs that usability experts bring to evaluations.

Nielsen (1992) investigated the effects of evaluator expertise on the number of problems identified using HE and claims that "usability specialists were much better than those without usability expertise by finding usability problems with heuristic evaluation." Gray and Salzman claim this study lacks *conclusion validity* because it is not clear what the effect of HE was on the evaluator's ability to find usability problems or how many problems evaluators would find through expert review without recourse to HE's evaluation resources. They suggest the data supported the more modest claim that "experts named more problems than non-experts."

Gray and Salzman found that many studies presented conclusions that were not supported by the data. They do not argue against presenting advice based on experience rather than experimental evidence, but argue that the source of such advice should be made explicit. Since their critique, our knowledge of the range of evaluation resources that influences evaluator performance has significantly increased, both through attempts to avoid the methodological flaws identified by Gray and Salzman and by more recent realistic case studies of usability work. The former research tactics (avoiding methodological flaws) have exposed the impact of problem reporting formats, matching and merging procedures, and

task selection resources on UIM performance. As only HW provides any such resources (task selection), no UIM can be assessed in isolation. It must be supplemented by other evaluation resources, which may themselves have a major impact on evaluator performance (as in Cockton et al. 2003). Studies of usability work have revealed an even wider range of evaluation resources that have far more impact on evaluation outcomes than resources provided by UIMs. Usability work requires evaluation resources that are rarely provided by UIMs, which thus cannot be compared by comparing the outcomes of evaluations, since these will be confounded by many evaluation resources in the study context. However, it is possible to manipulate a single resource such as a problem report format and hold other evaluation resources fairly constant (through randomization). Such focused studies, rather than crude comparisons of UIMs, are one important way forward in HCI research on evaluation methods. The methodological challenges identified by Gray and Salzman simply cannot be overcome, but there have been some useful spin offs from continued attempts to rigorously compare UIMs.

56.6 CHOOSING AND USING USABILITY INSPECTION METHODS

Based on the discussion so far, it should be clear that choosing, combining, and using UIMs are creative processes; simply picking an inspection method because it is easy to use or applicable early in a project is insufficient and likely to fail. Next we provide some guidance on planning usability inspections, but temper this with an overview of the need to select resources that can (and will) be provided by a specific project.

56.6.1 Planning Usability Inspections

The DCRs identified as a result of exploring the DARe model do not simply enumerate the groups of resources used in usability inspection; DCRs are also a checklist for things to think through when preparing for an inspection. For instance, many usability inspections fail because evaluators know too little about the product, users' tasks, and the domain to be supported. Many studies have documented how knowledge about these things may improve inspection, to the extent that some evaluation approaches successfully trade in the need for knowledge about interaction and technical issues for domain knowledge (e.g., Følstad 2007). In planning inspections, usability evaluators need to consider whether knowledge resources concerning products, users' tasks, and the domain are available and sufficient for the inspection. Ideally, project managers and usability specialists need to plan support for inspections from the project outset to ensure that the necessary resources are in place to support high quality inspection. Such resources have multiple uses beyond supporting usability inspection. They also support planning of participant recruitment and selection of tasks (when appropriate) for user testing and also design activities such as authoring personas and scenarios. Planning for usability inspections should not start just before inspections are scheduled. Instead, resources required for all user-centered design activities need to be identified early in a project and sourced in good time. Otherwise, it is unreasonable to expect usability inspection to have high *downstream utility* (John and Marks 1997), which would be achieved by correcting, identifying, and understanding problems and recommending appropriate design change recommendations on this basis. Downstream utility cannot compensate for *upstream futility;* if evaluators are not adequately supported, then poor quality inspections result. In this sense, the tactics underlying *discount methods* are misguided. While a UIM can be reduced to 10 heuristics, advice on using multiple evaluators and a very rough two pass procedure, the effectiveness of UIMs such as HE is substantially compromised by deficiencies across all DCRs.

The DCRs revealed via the DARe model indicate that many different resources enter usability inspection. This is in agreement with the recommendation that multiple analysts may improve inspection method thoroughness (Nielsen and Landauer 1993). The DARe model explains why multiple evaluators find more problems when the UIM and environment remain the same. Quite simply, as more evaluators are added, more problem discovery resources are added to the inspection, hence more problems are found. However, the DARe model also leads us to considering the impact of multiple evaluators on analysis resources. If evaluators are inclined to not reject problems at all or if they are unduly confident in confirming problems, then multiple evaluators will not only be collectively more thorough, and thus find more problems, but they will also be collectively less valid and thus predict more false positives (Cockton and Woolrych 2002).

The more recent focus on *usability work* has greatly extended DCRs with a broader understanding of project-specific resources, as discussed earlier. These too provide another checklist for planning evaluation support within software and hardware development projects. For instance, the *goal* of the evaluation is a consideration too frequently stepped over in planning. The goal may be a formative or a summative evaluation; a global evaluation or a targeted assessment of a feature; negotiated among stakeholders or relatively open; and to identify solutions or pinpoint difficulties. Being explicit about the goal and configuring usability work to reflect it is crucial: it especially supports the logical phases of confirmation or elimination of candidate problems and problem reporting. Furthermore, the goals of evaluation need to focus on design purpose (Cockton 2005). This can be encouraged by report formats that include a focus on business impact or other forms of design purpose (e.g., social or personal impact), as illustrated by Hornbæk and Frøkjær (2008b). As already noted, reporting formats and merging and matching procedures should also be considered. Again, when planning an inspection, considering these and other resources would provide useful input to planning. Throwing evaluation resources together in a rush is a recipe for ineffective poor quality inspections

Planning for usability inspection is thus best considered as one aspect of overall project management. While usability inspections can be "slipped in" at short notice into a project schedule and will always deliver some value, it is better to plan

for them within an overall user-focused process, balancing other concerns such as business impact (e.g., for e-commerce sites) as appropriate. DCRs, including project-specific resources identified above, provide a checklist that can support planning. Advance planning allows time for careful consideration of heuristic sets, with project-specific amendments and extensions as necessary. It also allows time for careful consideration of severity scales, which must be closely related to design purpose. The severity of a usability problem is directly related to its impact on achieving the intended purpose for an interactive system or device. Severity problems increase as usability problems progressively degrade the value or worth that either users can gain from interaction or the achievable worth for other stakeholders such as website owners or other project. Without project-specific severity scales in place, evaluators must use *ad hoc* generic scales that are close to meaningless.

56.6.2 Matching Approach Resources to Project Resources

Project planning and resourcing will always be constrained by a range of internal and external considerations. For example, client needs and expectations will control what can and cannot be considered as evaluation resources. The first two authors have worked on a range of usability consultancies where the client insisted on exclusive use of user testing, even where it was clear from a brief "guerilla" inspection that much better value would arise for user testing if an inspection was followed by a set of design changes first. However, many clients find inspection methods too subjective and unscientific, even when evaluators can offer decades of combined usability expertise. This mirrors trends from surveys of usability professionals. HE was the most used usability method in a paper questionnaire-based survey of 111 usability professionals in 1999 (Rosenbaum, Rohn, and Humburg 2000). However, a more recent web survey of 83 usability professionals, with a detailed follow up for 16 respondents (Venturi, Troost, and Jokela 2006), now ranks inspection methods (38% of respondents use during design) well below discount user testing (quick and dirty usability tests used by 53%—in contrast to 70% and 65%, respectively, in the 1999 survey). Although surveys must be compared with great caution, UIMs may no longer be the predominant method in usability practice. User-based methods are increasingly preferred because, once selected and briefed, individual test participants generate many complete usage scenarios. Hence, user testing can expose problems through user behaviors that are very hard to anticipate. However, testing only reveals a wide possible range of usability problems if users stress the system with complex interactions across all features. It is thus possible to anticipate user difficulties that may not emerge during testing, unless specific efforts are made to flush out all predicted problems (Woolrych, Cockton, and Hindmarch 2004). However, UIMs are still in widespread use, sometimes because the cost and logistical challenges of recruiting test participants leads some clients to rule out user testing, placing the whole usability evaluation load on UIMs (including expert inspection).

One of the most important resources for evaluation is a clear sense of design purpose and project vision. Such a resource can be frustratingly rare, but without it makes it very difficult to devise adequate severity scales, nor is there a sound basis for prioritizing design changes on the basis of well grounded criteria. For commercial systems, it is important to relate purpose and priorities to the business context. Without such resources, there is limited support for considering business impact during usability inspections. As with other key evaluation resources, in the absence of information and/or adequate project leadership, usability specialists need to find ways to compensate for lack of support from project management. Even when there is clear vision and good project leadership, not all recommended resources for UIMs can be provided. For example, multiple evaluators have been shown to improve thoroughness for HE and other UIMs, but these evaluators have to be sourced, and finding 5–8 expert evaluators with HCI expertise will not be possible for most projects. However, usability leads may be able to provide some training in UIMs for other development staff. Alternatively, a Pluralistic Walkthrough (PW) can be chosen, with specialists with roles such as marketing, training, accessibility, support, or documentation providing alternative complementary expertise to the usability specialist.

Inspections need to take place as soon as any aspect of a design can be evaluated. UIMs provide their best value early in the development cycle. The longer design choices are adhered to, the harder it is to make changes as a result of an inspection, which is sometimes why clients require user testing, since more credible but more costly data may be needed to establish the need to undo existing expensively implemented design decisions. A common exception here are those web sites (e.g., e-commerce websites) that are often redesigned at set intervals, with a corresponding expectation that most existing design decisions can be revisited and revised if necessary.

56.7 FUTURE OF USABILITY INSPECTIONS: MAKING DIFFERENCES TO USABILITY WORK

UIMs are approaches that offer bundles of resources that can be adapted and complemented to support usability work. Thus, while there is much evidence of difficulties in UIM use, especially as regards evaluation quality, most problems associated with the use of UIM resources such as heuristics and walkthrough procedures are avoidable. Our view remains that, used with care, UIMs remain key approaches in the usability toolbox: "The challenge is to improve all HCI methods, so that discount methods are less discounted and 'full strength' methods can be applied in more contexts" (Cockton and Woolrych 2002, p. 29). The most appropriate use of UIMs is to drive design iterations, rather than use for summative evaluation, benchmarking, or competitor analysis. UIMs are cheap to apply and are seen as low cost and low skill (but more resources and expertise can greatly improve evaluation performance). They have been applied to many commercial designs by practitioners. UIMs can be used before a testable prototype has been implemented and can be iterated without exhausting

or biasing a group of test participants. UIMs can also be a planning resource for user testing, which can be designed to focus on predicted problems. Development resources may rule out user testing, leaving UIMs as the only possible approach, and some usability is always better than no usability.

UIMs remain important and their weaknesses can be mitigated. The main risks are missing serious usability problems and wasting development effort through poor fixes to mispredicted nonproblems (false positives). Different business models make different demands; thus, UIM errors are more costly in some development contexts than others are. In some contexts, successful predictions may always be worthwhile despite a flood of false positives, as fixed problems translate into savings on support costs and attractive new features. When users buy software, most must struggle on, unlike visitors to free e-commerce sites, yet mainstream UIMs do not cover their complete user experience, especially for affective issues such as trust, comfort, and brand image. Hence, user testing is vital to eliminate any severe problems; otherwise, money will be lost.

Novice evaluators should learn how to configure and augment UIMs before practice on familiar systems and usage contexts to establish quickly their scopes and accuracies. No UIM provides contextual information or understandings of users and tasks (Cockton and Woolrych 2002). Fortunately, HCI professionals apply contextual research (Venturi, Troost, and Jokela 2006) even though some may not accept that contextual resources are not part of UIMs (Manning 2002). Novice UIM users need to be coached to properly augment UIM resources with an appropriate complement of DCRs and make full use of available project resources. In particular, evaluators must know about and understand the system they are inspecting. Some UIM developers disagree, preferring to keep analysts in untainted ignorance. They aim to induce user empathy, but this can only be truly grounded in contextual resources. Ungrounded beliefs about users and usage contexts can be very unreliable (and often insult real users' intelligence). If evaluators begin inspection ignorant of both the usage context and the system being evaluated, then the result is not better empathy with the user but incorrect claims that features are missing or beyond users. While there may be evidence of usability problems in such evaluator errors, they are still errors. Properly informed evaluators are more likely to note that a feature is hard to find (rather than absent) or that a design rationale has overlooked some contextual criteria. Such problem predictions are far preferable to bogus ones, which are major risks in UIM usage. In short, evaluator preparation is crucial to successful UIM usage.

By properly configuring and augmenting UIMs, evaluators can improve the quality of discovery and analysis resources. Knowledge of expected contexts of use, human capabilities, and key properties of HCI are critical. For example, knowledge of display-based interaction may eliminate possible problems that overlook users' abilities to discover information and to explore interactive behaviors. Knowledge of human capabilities such as visual attention may either confirm possible problems (e.g., key information in the wrong place) or eliminate them (e.g., misleading information in the wrong place).

Evaluators thus need to be expected to work to make UIMs work. However, in the next few years, we expect to see a growth of in-depth case studies of evaluation practices and outcomes in the HCI literature. Usability specialists will be able to draw on these case studies to improve their practice. Note here that the aim of HCI research on UIMs is moving away from method assessment and comparison towards studies of usability work. The aim is less to improve UIMs per se, but instead to improve their usage and to do so by improving evaluators through improved intelligence on the inspection process. At some point, a critical mass of well designed and well executed case studies will support derivation of complex system models of usability work, from which hypotheses can be derived. Well focused experimental studies can then be designed to test these hypotheses. At this point, the scientific knowledge sought in initial comparisons of UIMs will be more likely, but it will not take the form of simple rankings of UIMs, but instead deep understandings of the impact on evaluation performance of specific configurations and combinations of evaluation resources. Until then, expertise in usability inspection is primarily the responsibility of evaluators. UIMs themselves cannot be sufficiently improved to guarantee success. However, imaginative use of specific resources such as report formats and specialized heuristics can have disproportionate impacts. The core knowledge for usability inspection does not, thus, concern "methods" but resources. Evaluators need to understand the bases for quality for each evaluation resource and put this knowledge to the best effect when configuring and combining resources for usability inspection.

We expect the next few years to open up new research approaches for UIMs (Woolrych, Hornbæk, Frøkjær and Cockton 2012), which will feed through into education and practice. These new approaches will firstly focus within methods on different types of resource (e.g., knowledge, scoping, procedural, analytic) and the impact of variations for a single resource, and secondly above the method level on usability *work activities*. Methods are simply the wrong unit of analysis for understanding evaluation practice. By focusing on methods, research has not paid enough attention to variations within use of the same method, and has thus attributed specific impacts to methods that are actually impacts of specific resources. The impact of different formats and content for problem report formats has clearly demonstrated that specific resources can have disproportionate impacts on evaluation performance. Also, by focusing on individual evaluation methods, research has not paid enough attention to how sets of methods are selected, adapted and combined during usability work. What matters in usability work are the outcomes of specialist investigations and advice, and that is the result of complex interactions within a mix of contextual factors. Methods, or rather the bundling of resources into approaches, are only aspect of this mix. Methods have to be related to the big picture in interaction design work. We expect that this big picture will be a major focus of HCI research in the current decade, resulting in a much more sophisticated understanding of usability work and the role of methods within this.

REFERENCES

Andre, T. S., H. R. Hartson, S. M. Belz, and F. A. McCreary. 2001. The user action framework: A reliable foundation for usability engineering support tools. *Int J Hum Comput Stud* 54(1):107–36.

Bastien, J. M. C., and D. L. Scapin. 1995. "Evaluating a user interface with ergonomic criteria." *Int J Hum Comput Interact* 7(2):105–21.

Bias, R. G. 1994. The pluralistic usability walkthrough: Coordinated empathies. In *Usability Inspection Methods*, ed. J. Nielsen and R. L. Mack. New York: John Wiley and Sons.

Capra, M., and T. Smith-Jackson. 2005. *Developing Guidelines for Describing Usability Problems (No. ACE/HCI-2005-002)*. Blacksburg, VA: Virginia Tech, Assessment and Cognitive Ergonomics Laboratory & Human–Computer Interaction Laboratory.

Cockton, G. 2005. A development framework for value-centred design. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, ed. G. C. van der Veer and C. Gale, 1292–195. New York: ACM Press.

Cockton, G., and D. Lavery. 1999. A framework for usability problem extraction. In *Proceedings of Interact '99*, ed. M. A. Sasse, and C. Johnson, 344–52. Amsterdam, The Netherlands: IOS Press.

Cockton, G., and A. Woolrych. 2001. Understanding inspection methods: Lessons from an assessment of heuristic evaluation. In *People and Computers XV: Interaction without Frontiers*, eds. A. Blandford and J. Vanderdonckt, 171–91. London: Springer-Verlag.

Cockton, G., and A. Woolrych. 2002. Sale must end: Should discount methods be cleared off HCI's shelves? *Interactions* 9(5):13–8.

Cockton, G., and A. Woolrych. 2009. *Comparing UEMs: Strategies and Implementation*. Final report of COST-294 working group 2. http://141.115.28.2/cost294/upload/533.pdf. (Accessed November 30, 2011).

Cockton, G., A. Woolrych, L. Hall, and M. Hindmarch. 2003. Changing analysts' tunes: The surprising impact of a new instrument for usability inspection method assessment. In *People and Computers XVII: Designing for Society*, eds. P. Palanque, P. Johnson, and E. O'Neill, 145–62. London: Springer-Verlag. John Long Award for Best Paper.

Cockton, G., A. Woolrych, and M. Hindmarch. 2004. Reconditioned merchandise: Extended structured report formats in usability inspection. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, ed. E. Dykstra-Erickson and M. Tscheligi, 1433–6. New York: ACM Press.

Connell, I. W., and N. V. Hammond. 1999. Comparing usability evaluation principles with heuristics: Problem instances vs. problem types. In *IFIP INTERACT '99: Human–Computer Interaction*, eds. M. A. Sasse and C. Johnson, 621–9. Amsterdam, The Netherlands: IOS Press.

Cook, T. D., and D. T. Campbell. 1979. *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Chicago: Rand McNally.

Cuomo, D. L., and C. D. Bowen. 1992. Stages of user activity model as a basis for user-system interface evaluations. In *Proceedings of the Human Factors Society 36th Annual Meeting, Human Factors Society*, 1254–8. Santa Monica, CA: Human Factors Society.

Cuomo, D. L., and C. D. Bowen. 1994. Understanding usability issues addressed by three user-system interface evaluation techniques. *Interact Comput* 6(1):86–108.

Frøkjær, E., and K. Hornbæk. 2002. Metaphors of human thinking in HCI: Habit, stream of thought, awareness, utterance, and knowing. In *Proceedings of HF2002/OzCHI 2002 (CD-Rom)*, eds. R. Kuchinsky, L. Johnson, and F. Vetere. Australia: CHISIG.

Frøkjær, E., and K. Hornbæk. 2008. Metaphors of human thinking for usability inspection and design. *ACM Trans Comput Hum Interact* 14(4):1–33.

Furniss, D. 2008. *Beyond Problem Identification: Valuing Methods in a 'System of Usability Practice.'* London: University College London.

Følstad, A. 2007. Work-domain experts as evaluators: Usability inspection of domain-specific work-support systems. *Int J Hum Comput Interact* 22(3):217–45.

Gray, W. D., and M. Salzman. 1998. Damaged merchandise? A review of experiments that compare usability evaluation methods. *Hum Comput Interact* 13(3):203–61.

Green, T. R. G. 1991. Describing information artifacts with cognitive dimensions and structure maps. In *Proceedings of the HCI'91 Conference on People and Computers VI*, eds. D. Diaper and N. Hammond, 297–315. Cambridge, UK: Cambridge University Press.

Green, T. R. G., and D. Benyon. 1995. Displays as data structures: Entity-relationship models of information artifacts. In *Proceedings of INTERACT'95: IFIP TC13 Fifth International Conference on Human–Computer Interaction*, eds. K. Nordby, P. Helmersen, D. Gilmore, and S. Arnesen, 55–60. London: Chapman & Hall.

Green, T. R. G., and D. Benyon. 1996. The skull beneath the skin: Entity-relationship models of information artefacts. *Int J Hum Comput Stud* 44(6):801–28.

Green, T. R. G., and M. Petre. 1996. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *J Visual Lang Comput* 7(2):131–74.

Hartson, H. R., T. S. Andre, and R. C. Williges. 2001. Criteria for evaluating usability evaluation methods. *Int J Hum Comput Interact* 13(4):373–410.

Hertzum, M., and N. E. Jacobsen. 2001. The evaluator effect: A chilling fact about usability evaluation methods. *Int J Hum Comput Interact* 13(1):421–43.

Hornbæk, K., and E. Frøkjær. 2002. Evaluating user interfaces with metaphors of human thinking. In *Proceedings of User Interfaces For All*, Paris, France, October 23–25, also in Springer Lecture Notes in Computer Science, Vol. 2615, 486–507. Berlin: Springer.

Hornbæk, K., and E. Frøkjær. 2004. Usability inspection by metaphors of human thinking compared to heuristic evaluation. *Int J Hum Comput Interact* 17(3):357–74.

Hornbæk, K., and E. Frøkjær. 2006. What kinds of usability-problem description are useful to developers? In *Human Factors and Ergonomic Society's Annual Meeting*, 2523–7. Santa Monica, CA: Human Factors Society.

Hornbæk, K., and E. Frøkjær. 2008a. Comparison of techniques for matching of usability problem descriptions. *Interact Comput* 20(6):505–14.

Hornbæk, K., and E. Frøkjær. 2008b. Making use of business goals in usability evaluation: An experiment with novice evaluators. In *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, 903–12. New York: ACM Press.

Howarth, J., T. S. Andre, and R. Hartson. 2007. A structured process for transforming usability data into usability information. *J Usability Stud* 3(1):7–23.

Hvannberg, E. T., E. L.-C. Law, and M. K. Lárusdóttir. 2007. Heuristic evaluation: Comparing ways of finding and reporting usability problems. *Interact Comput* 19(2):225–40.

Jacobsen, N. E., M. Hertzum, and B. E. John. 1998. The evaluator effect in usability tests. In *Human Factors in Computing Systems CHI'98 Summary*, eds. C.-M. Karat and A. Lund, 255–6. New York: ACM Press.

Jeffries, R. 1994. Usability problem reports: Helping evaluators communicate effectively with developers. In *Usability Inspection Methods*, eds. J. Nielsen and R. L. Mack, 273–94. New York: John Wiley and Sons.

Jeffries, R., J. R. Miller, C. Wharton, and K. M. Uyeda. 1991. User interface evaluation in the real world: A comparison of four techniques. In *Proc. CHI'91 Conf. on Human Factors in Computing Systems*, eds. S. P. Robertson, G. M. Olson, and J. S. Olson, 119–24. New York: ACM.

John, B. E., and S. J. Marks. 1997. Tracking the effectiveness of usability evaluation methods. *Behav Inf Technol* 16(4/5):188–202.

John, B. E., and H. Packer. 1995. Learning and using the cognitive walkthrough method: A case study approach. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, eds. I. Katz, R. Mack, and L. Marks, 429–36. New York: ACM Press.

Lavery, D., and G. Cockton. 1996. Iterative development of early usability evaluation methods for software visualisations. In *Proceedings of the 6th Workshop of Empirical Studies of Programmers*, eds. W. D. Gray and D. A. Boehm-Davis, 275–6. Ablex. Glasgow, UK: Glasgow University.

Lavery, D., G. Cockton, and M. Atkinson. 1996a. *Heuristic Evaluation: Usability Evaluation Materials*. Technical Report TR-1996-15, Department of Computing Science. Glasgow, UK: University of Glasgow.

Lavery, D., G. Cockton, and M. Atkinson. 1996b. *Cognitive Dimensions: Usability Evaluation Materials*. Technical Report TR-1996-17, Department of Computing Science. Glasgow, UK: University of Glasgow.

Lavery, D., G. Cockton, and M. P. Atkinson. 1997. Comparison of evaluation methods using structured usability problem reports. *Behav Inf Technol* 16(4):246–66.

Lewis, C., P. Polson, C. Wharton, and J. Rieman. 1990. Testing a walkthrough methodology for theory-based design of walk-up—and-use interfaces. In *Proc. CHI'90 Conf. on Human Factors in Computing Systems*, eds. J. Carrasco and J. Whiteside, 235–42. New York: ACM Press.

Lewis, C., and C. Wharton. 1997. Cognitive walkthroughs. In *Handbook of Human–Computer Interaction*, 2nd ed., eds. M. Helander, T. K. Landauer, and P. Prabhu, 717–32. New York: Elsevier.

Lindgaard, G., and J. Chattratichart. 2007. Usability testing: What have we overlooked? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1415–24. New York: ACM Press.

Mankoff, J., A. K. Dey, G. Hsieh, J. Kientz, M. Ames, and S. Lederer. 2003. Heuristic evaluation of ambient displays. CHI Letters, CHI 2003. *ACM Conf Hum Fact Comput Syst* 5(1):169–76.

Manning, H. 2002. Reflections: Must the sale end? *Interactions* 9(6):56. ACM.

Medlock, M., D. Wixon, M. McGee, and D. Welsh. 2005. The rapid iterative test and evaluation method: Better products in less time. In *Cost-Justifying Usability: An Update for the Information Age*, eds. R. Bias and D. Mayhew, 489–517. Morgan Kaufman.

Molich, R., M. R. Ede, K. Kaasgaard, and B. Karyukin. 2004. Comparative usability evaluation. *Behav Inf Technol* 23(1):65–74.

Molich, R., and J. Nielsen. 1990. Improving a human–computer dialogue. *Commun ACM* 33(3):338–48.

Muller, M. J., A. McClard, B. Bell, S. Dooley, L. Meiskey, J. A. Meskill, R. Sparks, and D. Tellam. 1995. Validating an extension to participatory heuristic evaluation: Quality of work and quality of life. In *Proc. ACM CHI'95 Conference on Human Factors in Computing Systems (Conference Companion)*, eds. I. Katz, R. Mack, and L. Marks, 115–6. New York: ACM Press.

Nielsen, J. 1992. Finding usability problems through heuristic evaluation. In *Proc. ACM CHI'92 Conf.*, eds. P. Bauersfeld, J. Bennett, and G. Lynch, 373–80. New York: ACM press.

Nielsen, J. 1993. *Usability Engineering*. San Francisco: Morgan Kaufmann.

Nielsen, J. 1994a. Enhancing the explanatory power of usability heuristics. In *Proc. CHI'94 Conference on Human Factors in Computing Systems*, eds. B. Adelson, S. Dumais, and J. Olson, 152–8. New York: ACM Press.

Nielsen, J. 1994b. Heuristic evaluation. In *Usability Inspection Methods*, eds. J. Nielsen and R. L. Mack, 25–62. New York: John Wiley & Sons.

Nielsen, J., and T. K. Landauer. 1993. A mathematical model of the finding of usability problems. In *Proc. INTERCHI'93 Conf. on Human Factors in Computing Systems*, eds. S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. White, 206–13. New York: ACM Press.

Nielsen, J., and R. Molich. 1990. Heuristic evaluation of user interfaces. In *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, eds. J. Carrasco and J. Whiteside, 249–56. New York: ACM Press.

Norman, D. A. 1986. Cognitive engineering. In *User Centered System Design: New Perspectives on Human–Computer Interaction*, eds. D. A. Norman and S. W. Draper, 31–61. Hillsdale, NJ: Lawrence Erlbaum Associates.

Nørgaard, M., and K. Hornbæk. 2006. What do usability evaluators do in practice? An explorative study of think-aloud testing. In *ACM Conference on Designing Interactive Systems*, 209–18. New York: ACM Press.

Polson, P. G., and C. H. Lewis. 1990. Theory-based design for easily learned interfaces. *Hum Comput Interact* 5(2–3):191–220.

Polson, P. G., C. Lewis, J. Rieman, and C. Wharton. 1992. Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *Int J Man Mach Stud* 36(5):741–73.

Rieman, J., S. Davies, D. C. Hair, M. Esemplare, P. Polson, and C. Lewis. 1991. An automated cognitive walkthrough. In *Proc. ACM CHI'91 Conf. on Human Factors in Computing Systems*, eds. S. P. Robertson, G. M. Olson, and J. S. Olson, 427–8. New York: ACM Press.

Rosenbaum, S. 2008. The future of usability evaluations: Increasing impact on value. In *Maturing Usability: Quality in Software, Interaction and Value*, eds. E. L. C. Law, E. Hvannberg, and G. Cockton, 344–78. London: Springer.

Rosenbaum, S., J. A. Rohn, and J. Humburg. 2000. A toolkit for strategic usability: Results from workshops, panels, and surveys. In *Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems*, eds. R. Little and L. Nigay, 337–44. New York: ACM Press.

Rowley, D. E., and D. G. Rhoades. 1992. The cognitive jogthrough: A fast-paced user interface evaluation procedure. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, eds. P. Bauersfeld, J. Bennett, and G. Lynch, 389–95. New York: ACM Press.

Scapin, D. L. 1990. Organizing human factors knowledge for the evaluation and design of interfaces. *Int J Hum Comput Interact* 2(3):203–29.

Scapin, D. L., and J. M. C. Bastien. 1997. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behav Inf Technol* 16(4/5):220–31.

Sears, A. 1997. Heuristic walkthroughs: Finding the problems without the noise. *Int J Hum Comput Interact* 9(3):213–34.

Sears, A., and D. Hess. 1999. Cognitive walkthroughs: Understanding the effect of task description detail on evaluator performance. *Int J Hum Comput Interact* 11(3):185–200.

Somervell, J., and D. S. McCrickard. 2005. Better discount evaluation: Illustrating how critical parameters support heuristic creation. *Interact Comput* 17(5):592–612.

Spencer, R. 2000. The streamlined cognitive walkthrough method, working around social constraints encountered in a software development company. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 353–9. New York: ACM.

Spool, J., and W. Shroeder. 2001. Testing web sites: Five users is nowhere near enough. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, 285–6. New York: ACM.

Suchman, L. 1987. *Plans and Situated Actions: The Problem of Human-Machine Communication*. New York: Cambridge University Press.

Sy, D. 2007. Adapting usability investigations for agile user-centered design. *J Usability Stud* 2(3):112–32.

Theofanos, M., and W. Quesenbery. 2005. Towards the design of effective formative test reports. *J Usability Stud* 1(1):27–45.

Uldall-Espersen, T., E. Frøkjær, and K. Hornbæk. 2008. Tracing impact in a usability improvement process. *Interact Comput* 20(1):48–63.

Venturi, T., J. Troost, and T. Jokela. 2006. People, organizations, and processes: An inquiry into the adoption of user-centered design in industry. *International Journal of Human-Computer Interaction*, 21(2):219–238.

Wharton, C., J. Rieman, C. Lewis, and P. Polson. 1994. The cognitive walkthrough: A practitioner's guide. In *Usability Inspection Methods*, eds. J. Nielsen and R. L. Mack, 105–40. New York: John Wiley & Sons.

Wixon, D. 2003. Evaluating usability methods: Why the current literature fails the practitioner. *Interactions* 10(4):28–34.

Woolrych, A. 2001. *Assessing the Scope and Accuracy of the Usability Inspection Method Heuristic Evaluation*, MPhil, University of -Sunderland. http://osiris.sunderland.ac.uk/,cs0awo/downloadable%20documents.htm (accessed 21/12/05).

Woolrych, A., and G. Cockton. 2000. Assessing heuristic evaluation: Mind the quality, not just percentages. In *Proceedings of British HCI Group HCI 2000 Conference*, Vol. 2, eds. S. Turner and P. Turner, 35–6. London: British Computer Society.

Woolrych, A., and G. Cockton. 2002. Testing a conjecture based on the DR-AR model of usability inspection method effectiveness. In *People and Computers XVI: Memorable yet Invisible*, Vol. 2, eds. H. Sharp, P. Chalk, J. LePeuple, and J. Rosbottom, 30–3. London: British Computer Society.

Woolrych, A., G. Cockton, and M. Hindmarch. 2004. Falsification testing for usability inspection method assessment. In *Proceedings of HCI 2004*, Vol. 2, eds. A. Dearden, and L. Watts, 137–40. Bristol, UK: Research Press International.

Woolrych, A., G. Cockton, and M. Hindmarch. 2005. Knowledge resources in usability inspection. In *Proceedings of HCI 2005*, Vol. 2, eds. L. Mackinnon, O. Bertelsen and N. Bryan-Kinns, 15–20.

Woolrych, A., K. Hornbæk, E. Frøkjær, and G. Cockton. 2012. Ingredients and meals rather than recipes: A proposal for research that does not treat usability evaluation methods as indivisible wholes. *International Journal of Human-Computer Interaction* 27(10):940–70.

Young, R. M., and T. Simon. 1987. Planning in the context of human–computer interaction. In *People and Computers III*, eds. D. Diaper and R. Winder, 363–70. Cambridge, UK: Cambridge University Press.