Kasper Hornbæk, Per Ola Kristensson, and Antti Oulasvirta. Upcoming 2024. Introduction to Human-computer Interaction. Oxford University Press.

Part VIII.

Evaluation

40. Introduction to Evaluation

HCI is human centered. In other words, it takes an interest in the people who use computers. An essential challenge therein lies in assessing whether an interactive system will be or is 'good' for people, works as intended, and whether adverse effects will occur. We must ensure that the systems are practical, usable, and accessible and that they can deliver the value to the people its designers imagined. In HCI, such assessments are called *evaluations*. This part describes how to conduct evaluations of interactive systems.

In general, *evaluation* refers to the attribution of value. An evaluation states whether something is 'good' or 'bad', or if it 'fails' or is 'acceptable'. To arrive at such a judgment for an interactive system, one must assess a design against some *evaluation criteria*, some yardstick. These allow us to conclude in some definite way how good the system is.

Evaluation in this sense is different from the common-sense understanding of the word. Introspective opinions from the developer do not suffice because they – in most cases – are not a valid estimate of value offered to end-users. It is unwise – and often ethically untenable – to base an evaluation on self-reflection, as such information is inherently biased and unrepresentative. Evaluation methods should be systematic so that their results can be trusted, replicated. and scrutinized by others.

Evaluations have played a central role in HCI through its history. They have taught a lot about which design solutions work and in which way they do not work. For instance,

- Consolvo et al. [165] developed a mobile system to encourage physical activity and combat a primarily sedentary lifestyle. They evaluated how 12 people used the system for three weeks as part of their daily life. This evaluation showed us what types of physical activity the participants performed and what activities the system could and could not infer.

- Amershi et al. [19] collected 18 guidelines for a successful collaboration between humans and AI. They show how practitioners can use the guidelines to evaluate systems that rely on artificial intelligence. For instance, a guideline called "Make clear why the system did what it did" helped practitioners identify many cases where the systems violated the rule.

- The paper box below shows a classic HCI evaluation, which used experiments and quantitative measures of usability to improve a system called Superbook over several iterations. This evaluation helped improve the Superbook from being a reasonable manual to being better than paper manuals.

Evaluation is related to other parts of the book, but distinct. User research (Part III consists in obtaining concrete insights about *particular* users, their activities, their needs,

wants, and context of use. These insights are not evaluative and are typically collected before anything is designed. In contrast, evaluation is about evaluating the value of an interactive system for end-users. It is done after at least some implementation of an interactive system has been produced. In other words, evaluation is about how good an interactive system is, whereas user research is about informing what a good system might be. Somewhat confusingly, some empirical methods may be used for either purpose. An interview (see Chapter 11), for instance, may be used to understand users' activities and to evaluate a system; a think-aloud study (see Chapter 42) may be used both to learn about a user's work and to identify problems with usability. What differs is the *intention* of the researcher in applying the method. Evaluation is also part of engineering methods in HCI. Although there is great overlap with the methods covered here, Part VII discusses two specific types of evaluation: verification and validation.

It is worth being clear on *why* we evaluate systems. For these reasons, evaluation has become part of most models of human-centered system development. They include the following.

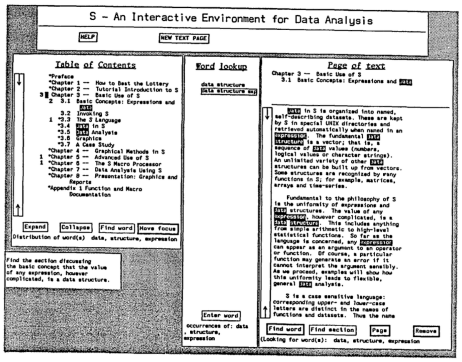
- It is nearly impossible to make systems correct in the first attempt. Gould and Lewis [280] pointed out that everyone builds a prototype. Some evaluate it one or more times, and some simply deliver it to the customer or the marketplace.
- Evaluation is profitable. The literature on the return-on-investment of evaluations is unequivocal. When money is spent on an evaluation, it is returned many times [72].
- Evaluation can involve and engage users. Involving stakeholders in the design and evaluation of interactive systems leads to better adoption of the systems and helps keep "continuous focus on users and their tasks" [280].
- Early evaluation helps prevent poorly thought-out or developed ideas from being designed and introduced to people. Again, this saves resources. For comparison, computer scientists and engineers often check and test that the systems store data reliably, do not crash, and give accurate results. However, they rarely consider the user perspective.
- Lack of evaluation can negatively reflect on designers and their organizations. The Association of Computing Machinery's code of ethics states that a computing professional should "strive to achieve high quality in both the processes and products of professional work"[1].

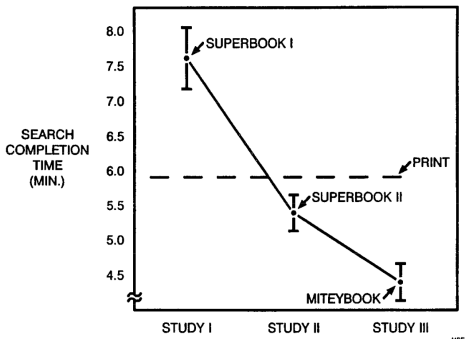
Next, we first discuss in more depth the objectives of evaluation and then turn to the yardsticks that may be used as the standard of evaluation. Based on this, we give an overview of evaluation methods and discuss their key quality characteristics.

[1] https://www.acm.org/code-of-ethics

Paper Example 40.0.1 : Can computers better support reading than paper?

Throughout the history of HCI, researchers have tried to understand why paper, as a material, is so good for reading and how to create interactive systems that could perform better than paper. The work on Superbook [215] is an example of iterative evaluation of a system. Superbook, a version of which is shown on the left, is a hypertext browsing system that provides access to content about statistics.





The evaluation of SuperBook consisted of three rounds of iterative software development and evaluation. Evaluations were carried out as comparisons of searches in SuperBook and printed manuals that offer the same information for tasks that required finding answers to a question and an essay writing task.

The evaluations revealed three important results. First, as shown in the figure to the right, the first version of SuperBook performs worse on question-answering than the printed manual. People's use of paper is flexible, and here even some of the brightest minds of HCI could not make a system that in one iteration performed better than the paper baseline. Second, through iterations, careful observations of interaction patterns and usability problems helped to improve the system. Third, the work shows that evaluations can use tasks that challenge users. In one test, participants wrote essays using either the SuperBook or the printed manual. An essay could, for instance, be about comparing three different functions of the statistics system described. Among other things, an expert in statistics assessed the essays. This example shows that an evaluation can address even complex tasks. One potential benefit is that challenging tasks can better tease out differences among systems.

40.1. Goals of Evaluation

Evaluations may be performed for a variety of reasons. In general, those reasons shape what is done in an evaluation, how it should be carried out, and how it is interpreted and reported. Therefore, being upfront about the goals of an evaluation helps to make the appropriate choices and plan the evaluation in the best possible way. Start with *why*.

A primary reason for performing the evaluation is to *improve an interactive system*. This is a frequent activity in the practical development of interactive systems. Most

Attribute	Measuring concept	Measuring method	Worst case	Planned level	Best case	Now level
Initial use	Conferencing task	No. of successful interactions in 30 min	1–2	3–4	8–10	?
Infrequent use	Task after 1-2 weeks disuse	% of errors	Equal to product B	50% better	0 errors	?
Preference over product B	Questionnaire score	Ratio of scores	Same as B		None prefer B	?

Table 40.1.: Examples of quantitative goals of a summative usability evaluation; adapted from Whiteside et al. [867], p. 799.

software companies have people trained to conduct human-centered evaluations of their software. These evaluations may identify features of the system that unexpectedly do not work well for a particular group of users, a particular task, or in a particular use situation. We may then change those features in a future version of the system. For this reason, evaluations for this purpose are called *formative* because they help shape a system. Formative evaluation is essential in the iterative development of a system because it provides information on what to improve in the upcoming iterations. As an example of improving a system with evaluation, icons in the early Xerox Star computer (early 1980s) were extensively evaluated [70]. Among other things, the evaluation investigated whether users could precisely associate a name with an icon and to what extent they found an icon easy or difficult to "pick out of a crowd". This evaluation helped to choose the set of icons used for the Star.

The other main reason for evaluations is to discover *how well an interactive system performs given some objective*. The goal here is not to inform the design but to ensure that the system satisfies the objectives. This objective may be part of a requirement specification, be included in a contract, be used to select a system for procurement or establish a superior design. This use of evaluation is often called a *summative* evaluation. As an example, Whiteside et al. [867] proposed the establishment of quantitative objectives for different aspects of an interactive system. Table 40.1 shows three examples of such objectives for a conference system. The specific methods used to investigate these objectives may be of any kind.

A related but important goal of the evaluation is *identify system features that work well*. This is not related to the two main evaluation goals but is important because it may help confirm that our expectations of a particular design have indeed been fulfilled. Moreover, pointing out the positive features of a design has been found to be a good way to make the evaluation stakeholders, for instance, developers of interactive systems, appreciate the evaluation. Therefore, usability reports often list positive findings, and meeting non-functional objectives (such as those in Table 40.1) may also be an important

finding. Evaluations can also teach us about factors outside the evaluation criteria. They may teach us about users and their tasks. For instance, a user in a think-aloud study may react to a task by saying "we do not usually do it that way", which is important information. Therefore, we are essentially getting information from the evaluation that is usually obtained through user research. An example of a discovery made in an evaluative study is given in the paper box.

Paper Example 40.1.1 : Discovery of cascading error patterns in speech input

Evaluative studies sometimes lead to discoveries. A study is run and an anomaly is found, some pattern that goes against expectations. By further analysis of process data – that is, measurements of what happened during interactions – potential causes are exposed.

For example, the discovery of cascading errors in speech input has been motivated for years by a finding made in 1998. Halverson et al. [304] studied three commercial automatic speech recognition systems from the late 1990s. Although the recognition accuracy was not as high as currently, the error type that the authors found remains relevant.

The authors found that the input of speech was not at the level of efficiency of keyboards. Even after extensive practice, the performance of users was inferior by a large margin. This finding went against the prevailing idea that speech input is 'natural' for users. Speaking to a computer is very different from speaking to another human. In particular, users make very different types of error.

Trying to learn more about this, the authors discovered a prevalent types of error: the cascading error. When a speech recognition error occurs, the user tries to fix the error by indicating which word needs to be fixed and then redictating the word. However, since the word was incorrectly recognized the first time, it still has a higher than average chance of being incorrectly recognized. Moreover, providing voice commands, like UNDO, to delete a word, can themselves be misrecognized. Shortcomings like these can lead to frustrating episodes in which a user tries to fix the same word over and over again in different ways. Even a single episode like this can destroy a user's average performance in an evaluative study. The discovery of cascading errors motivated several interaction techniques and speech recognition approaches specifically addressing this problem.

40.2. Yardsticks of Evaluation

As mentioned above, evaluations strive to *attribute value* to an interactive system. Such attribution requires some way of measuring value for end-users. However, to state that something is 'good' in the case of interaction is tricky. What is 'good'? If a system is very responsive, does it mean it is usable? It depends. If a user is skilled at using a poorly designed system, does it mean that the system is usable for that person? Again,

it depends. A defining characteristic of evaluation in HCI is that neither people nor technology can determine evaluation alone. What is 'valuable' emerges via interactions between people and technology.

The basic question in evaluation, then, is how we set our 'yardstick' against which we evaluate systems. Such yardsticks are typically derived from our view of what a good interaction is (see Part IV) or from the design objectives of user interfaces (see Part V). For instance, an interactive system is only useful if it is usable. Thus, we may use a particular task as a yardstick and assess a system as good if a large percentage of users (say 80%) can complete it.

Table 40.2 shows some examples of the kinds of interfaces that interactive systems may be evaluated against. Some of these are absolute. For instance, we may assess an interactive system on user satisfaction and use a questionnaire for which we have standard. Some ways of measuring user satisfaction provide reference values (e.g., the System Usability Scale gives a score between 0 and 100; typically, systems score 70 [40]). We may also set a goal for a system so that it is learnable over a certain period of time. Absolute yardsticks may be applied to a single system. Other yardsticks are relative. That is, an interactive system can only be considered valuable in comparison to another system, be it a competitor, an earlier version, or an alternative user interface. This is often accomplished by experiments.

To use a yardstick for evaluation, it should be *operationalized*. That means that general constructs (such as finding "no usability problems") need to be turned into a procedure that allows us to measure a system against the yardstick. For example, *usability* is often operationalized as a usability test where selected tasks are given to invited participants to complete, while their task completion time, errors, and satisfaction are measured. 'Error' can be measured in many ways, such as inaccurate presses, misconceptions, or entering faulty states in the system.

Finally, we need to make *conclusions* based on the data obtained. If a user fails three times in completing a task out of ten, can the system be considered 'usable'? Because the constructs are about good interaction, many of them have been covered in earlier parts of the book. For instance, they include usability, accessibility, autonomy, awareness, memory load, and many more.

Note that merely describing a person's interaction with a system does not constitute an evaluation. We may describe which commands people use to interact with a system, in which posture they interact, or with what type of content they engage. None of these, however, gives us information about whether the interaction is good or bad. For instance, even something as straightforward as time requires a standard: low time usage may be interpreted as lack of engagement, or it may be interpreted as high efficiency [346]. The valuation you make depends on your evaluation standard. However, descriptions may be useful in themselves. They may also, in combination with a yardstick of what is good or bad, be used for evaluation. Such a standard may be an optimal set of commands to compare against a biomechanically efficient pose, or the content of positive valence.

Yardstick	Definition	Example method
No (critical) usability problems	The system does not make the user err, confused, or give up a task	Think aloud study
Don't make the user think	The system should not "make me think" unnecessarily [431]	Think aloud study
Comply with guidelines	The system should comply with known characteristics of good systems as captured in guidelines.	Heuristic evaluation
Meets usability goals	The system should meet specified, quantitative goals for non-functional requirements	Summative usability test
Compares favorably to X	The system should be better than some baseline system X on some measure of usability, accessibility, or similar	Experiment
Compatibility with user practices	The system can be adopted, appropriated, and accepted to the everyday life of users	Field study
Reduce cost of maintaining system	Reduce calls to support center	Deployment study
Safety	The system does not subject anyone to an unacceptable level of risk	Multiple
Meets user requirements	Requirements specified based on user research are met	Multiple

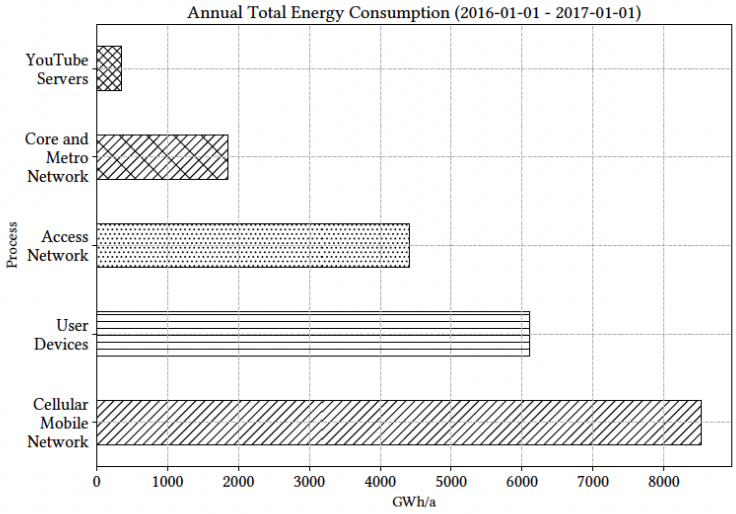
Table 40.2.: Examples of yardsticks against which to evaluate interactive systems.

Paper Example 40.2.1 : Evaluating Sustainability

Yardsticks for evaluation may be complicated to articulate and operationalize. This is because the evaluation may concern any aspect of when the interaction is good. To illustrate this complexity, let us consider how to evaluate whether a system is sustainable.

Sustainability has emerged as a topic in HCI since around 2007 [78]. Naturally, sustainability can act as a yardstick against which to evaluate a system. One such evaluation was done by Preist et al. [657], who were interested in how the digital service provides might accurately assess the greenhouse gas emissions associated with particular services. In particular, they were interested in quantifying the amount of greenhouse gas emissions resulting from a year of YouTube use.

The evaluation showed that YouTube use in 2016 created greenhouse gas emissions similar to those of Frankfurt or Providence.



Moreover, people often use YouTube only for audio. In the case where half of the music streaming on YouTube is audio only, it is possible to directly save 6% of the total greenhouse gas emissions. Moreover, the evaluation shows that the infrastructure for streaming content, in particular user's devices and the mobile network, contributes the most to emissions. According to Preist et al. [657], these network costs should be part of the sustainability evaluation.

40.3. Evaluation Methods

Evaluations are done with established systematic procedures to carry out them. Such procedures are called *evaluation methods*. Evaluation methods contain at least some yardstick for evaluation, a process for performing the evaluation, tools to support the

evaluation, and a standardized way of reporting the evaluation. Considerable research has gone into establishing good ways of evaluating systems that are valid, reliable, and useful as a way of giving input to the design of interactive systems.

Table 40.3 shows an overview of some key evaluation methods that will be covered in detail in subsequent chapters. These methods differ in some key ways. Often *analytic* and *empirical* methods are distinguished. In analytical evaluation methods, an evaluator compares an interface to guidelines, principles, or theories of good interaction. The assessment of the interface does not involve (actual) users. In empirical methods, users interact with the interactive system, and that is used as the basis for evaluation.

Another distinction between evaluation methods is whether they are performed in *laboratory* versus evaluations done in the *field*. Laboratory evaluations are done 'in vitro', away from the users' usual use context. They may therefore be done on interactive systems that are incomplete. The laboratory also allows for experimental control. Field evaluations are carried out during the actual use of a system. They offer fewer options for control. They emphasize realism [518]. Neither of these two approaches is superior to the other; each of them simply has different benefits and limitations. For a recap of realism, generalizability, and precision, and McGrath's arguments about the fallibility of all methods, see Part III.

Evaluation methods also differ with respect to what *representations of systems* they may be based on. Early representations of systems, including use cases, scenarios, and storyboards, can be evaluated. Paper prototypes, hi-fidelity prototypes, and interactive systems that have been used for years may also be evaluated by some techniques. Evaluation methods exist for all types of system representation.

As mentioned, the methods discussed in the user research part (Part III) may also form part of evaluations of interactive systems. Thus, all forms of interviews discussed in the chapter on interviews (**??**) can be part of the evaluation. The defining feature of evaluations is that they assess the value of systems relative to some yardstick. In themselves, interviews do not help to do that—we need some way to turn the content of interviews into a valuation of the interactive system.

40.3.1. Tailoring Evaluation Methods

Evaluation methods are not generic to all people, activities, technologies, and contexts. They may be tailored to particular instances of those, just as we learned that understanding people may draw on specific information about a group of people or a certain domain. For instance, Druin [210] discussed the strengths and challenges of involving 00 children in the testing of interactive systems. She argued that children can offer suggestions on tests that are surprising to adults. They must also be handled differently from adults during the test. The literature also contains adaptations for elderly users and people with disabilities [824].

Similarly, for specific technologies, we might also draw on evaluation approaches that are tailored to those technologies. For instance, researchers have developed guidelines that fit a range of technologies, including artificial intelligence [19], displays that are distributed in the environment [501], or groupware application [641].

Method	Definition	Pros	Cons
Heuristic evaluation	An analytic evaluation method in which evaluators go through an interface using a list of features of good user interfaces	Inexpensive, can be used on all representations of a system	False positives
Think aloud testing	Users verbalize what they think about while they solve tasks with an interactive system; the thinking aloud is analyzed to find usability problems.	Inexpensive, convincing	Short-term use, difficult to find some problems
Usability test	An evaluation of the usability of an interactive system with representative users doing representative tasks	Direct assessment of usability	Misses the broader context of use
Experiment	An experimental comparison of the usability of at least two user interfaces	High precision; clean comparisons	Limited realism
Deployment study	Measure evaluation criteria after deployment with real user	Problems are real	Expensive, may disrupt work

Table 40.3.: Central evaluation methods and their pros and cons.

Evaluation methods can also be tailored to particular activities. For example, the evaluation of games has seen the development of particular heuristics [805] and methods [521], the evaluation of mobile computing has seen the extensive use of treadmills as a way to simulate the demands of walking [64].

In all cases, the evaluation methods are improved by being more specific and tailored toward specific users, activities, contexts, or technologies. Unfortunately, little is known about how these customized evaluation methods perform relative to generic ones.

40.3.2. How to Choose an Evaluation Method?

At this stage, the reader might ask: Which of all these methods should we use in a given project? Due to the diversity in interactive systems, user goals, and use contexts, there is no silver bullet of method choice. Professional evaluators master a toolbox of methods and tailor them in a case-specific manner. Part of their considerations revolve around the goals of the evaluation relative to the pros and cons of each method. For instance, analytical methods work best for relatively simple designs and assume that experienced evaluators are available. Due to their high false negative rate, they should not be trusted for complex or safety-critical systems. For the novice evaluator, think-aloud testing is often a good choice.

40.3.3. Validity, Reliability, and Impact

As with user research methods (see Part III), evaluation methods raise several fundamental questions about the quality of their application. Some of these are similar to those discussed in the earlier part, but some are different.

The *validity* of an evaluation is about whether the evaluation result is the real value of the system. For instance, usability problems predicted by an evaluation method should be real problems for real users doing real tasks; otherwise, the evaluation is invalid.

In general, the results on the validity of different validation methods are mixed. However, a couple of findings stand out. First, numerous studies have shown that analytic evaluation can find a high proportion of problems that cannot be found in think-aloud studies. Second, even usability problems found in think-aloud tests might not be serious if the users that run into them in a test find a workaround that they can apply every time they subsequently face the problem.

The *reliability* of an evaluation refers to whether the findings of an evaluation would be changed with another set of evaluators or if it is repeated. If that is the case, the trustworthiness of findings is reduced, and it is unclear if action should be taken on the problems, as they might disappear if the evaluation was run again. Reliability has been the topic of much work on evaluation methods [e.g., 539, 375]. The Comparative Usability Evaluation (CUE) studies, for instance, have numerous times compared the performance of different evaluators or teams of evaluators on the same evaluations find markedly different usability problems[2].

[2]See more at https://www.dialogdesign.dk/cue-studies/

Impact is about ensuring that the evaluation results can be used for their purpose, in particular, with regard to formative evaluation. Impactful results will help change systems for the better by being convincing and offering concrete input or ideas on how to solve problems. John and Marks [380] compared the so-called persuasive power of usability evaluation methods. This refers to whether a developer, upon reading a description of a usability problem, actually changes the interactive system.

One of the advantages of usability tests, for example, is that they help convince developers that problems are significant. By seeing videos of users struggling, developers might be more easily persuaded that they must fix usability issues within a system.

40.4. Is Evaluation Needed?

Evaluation requires choosing appropriate yardsticks that best appraise the system against its intended effects. Such yardsticks vary greatly but may include benchmarking against other systems, avoiding usability problems, or increasing subjective satisfaction.

The assumption behind this is that evaluation, in particular empirical evaluation, is indispensable in HCI research and practice. However, several researchers have tried to nuance this view. It is worth emphasizing that these voices are primarily addressed toward HCI research and not HCI practice.

One example is an essay written for *CHI Fringe*[3] in 2003 entitled the "Tyranny of Evaluation" [1]. The essay does not say that evaluation is useless but laments the perceived sentiment (at the time) of top HCI publication venues being reluctant to accept HCI research papers proposing new ideas and systems without evaluation. The essay argues that it is frequently not possible to carry out sufficiently controlled experiments for the results to be reliable. Therefore, caution is advised and it is necessary to accept a plurality of ways of appreciating HCI research.

Five years later, a paper by Greenberg and Buxton [285], entitled "Usability Evaluation Considered Harmful (Some of the Time)" expanded on the essay. The argument in this paper is that often usability evaluations are carried out without much thought, merely because it is usually required or, as we have argued earlier, because there is an idea that evaluation is indispensable in HCI. However, thoughtless evaluations can harm scientific and practical progress. For example, evaluation does not help idea generation and, on the contrary, may undermine it. Evaluation does not help anticipate how people will adopt and adapt technology. Further, visions of future interfaces often necessitate a gradual evolution of imperfect prototypes. Such prototypes, particularly at the early stages, are unlikely to perform well in initial 'thoughtless' summative evaluations.

Another counter-argument to the need for evaluations is the view that interactive systems are objects of design or art. Their value is derived from the artist's or designer's vision and intuition; therefore, it is unnecessary to evaluate them. This view essentially means that the designer is giving up on being human-centered. To claim that a design

[3] CHI Fringe was a special ad-hoc track at CHI 2003. The first year the track was rather informal, and contributions were simply made available on individual authors' websites. Later, CHI Fringe became a formal track at CHI, which is currently referred to as alt.chi.

process has been human-centered places a burden of responsibility to demonstrate evidence that the results achievable with a system can be positive.

In summary, interactive systems are complex and simply assuming that they work as intended by their design is naive. Evaluation is about appraising whether a design meets its objectives. Evaluation is essential for the development of human-centered technology and indispensable for HCI research. Yet, there are cases where HCI-focused evaluation is not possible or needed. Evidence should be commensurate with claims. Whenever our design intends to have an effect on users, we should try to validate those claims via evaluation. Empirical evaluation is one way to do that but, as we learn in this Part, not the only way. However, if the objectives are not human-centered, but technology-focused or economical, the methods in this part may not be relevant. For example, if your work focuses on reducing rendering latency on a VR headset, technical measurements may suffice, assuming that other aspects affecting user experience are not changed. Yet, we believe that the fact that evaluation is hard and not infallible should not shun one away from it.

Summary

- Evaluation is necessary because systems are never perfect and because of the complexity of people, their activities, and physical, social, and organizational context.
- Evaluation methods have different strengths and weaknesses; they may be tailored to specific technologies and user groups
- Validity, realiability, and impact are key concerns for evaluation methods.

41. Analytical Evaluation Methods

Most evaluation methods we will discuss in this part are *empirical*; that is, their primary source of data is derived from measurements and observations of real people. Although empirical methods offer the gold standard for evaluation, they are labor-intensive, costly, and at times ethically problematic. Evaluators therefore started to ask if it would be possible to evaluate a design pre-empirically; that is, without involving users.

Analytic evaluation methods in HCI are a class of evaluation methods that do not require the collection of data on (real) users. Their purpose is to assess the usability of a design and expose probable errors based on good questions, rules of thumb, or models of performance prediction with a user interface. In general, analytical methods consist of (a) a process for performing the evaluation and (b) a set of resources to be used in the process. The resources help the evaluator predict what problems a user might encounter when using an interactive system, which parts of a user interface to focus on, or what the users' performance with the system might be. Importantly, some of these resources serve as a yardstick against which to compare a user interface or a design.

The main idea of an analytic method is simple to demonstrate: Let us consider the task of sending a message using your email app. First, take your mobile phone and open the messaging application. Then, decide on a recipient and a message that you would like to send. Then execute this task. While doing that, write down on paper all steps you needed to take; steps such as clicking, touching, scrolling, searching, deciding, etc. Now, repeat the sequence, but this time *purposefully introduce one error* somewhere in the sequence. For example, you could press the wrong button when you are supposed to send the message. What happened? Did the interface try to prevent you from making an error, for example, by disabling that button? Did it inform that there is an error? Did you easily recognize that you had made a mistake? If so, did the design help you recover from that error? These considerations exemplify one *heuristic evaluation guideline*, called "Help users recognize, diagnose, and recover from errors" (see section 41.1 for a comprehensive set of heuristics). Heuristic evaluation is one of the lightest-weight analytic methods, meaning that it is inexpensive and can be applied at all times throughout development.

Heuristics are *rules of thumb* that are often expressed as guidelines. They are imperatives that instruct how to design and how not to design: for example, *strive for consistency* [752]. HCI has a long history with guidelines, with guidelines developed for almost every interface technology, from computer terminals in the 1970s to AI systems in the 2020s. All major organizations have their design guidelines and associated design systems.

But where do the guidelines come from? Guidelines, as all analytical methods discussed in this chapter, are inspired by either the experience of professionals working in the field or theories of cognition. Most design guidelines are based on expert experience. They summarize what works and what does not. Johnson [383] is a recent example of

attempts to rethink guidelines from the cognition point-of-view. Johnson argues that proper rationale should be given for the guidelines. Many of them are directly rooted in an understanding of human cognition. Table 41.1 lists the guidelines based on Johnson's book *Designing with the Mind in Mind*.

Analytical evaluation methods, in general, are used for three purposes [32]:

1. In design, they are used to identify potential usability problems so that they can be rectified in design before deployment; this is a form of formative evaluation. For example, Langevin et al. [447] developed a heuristic evaluation method for conversational agents. The method can be used to identify potential usability problems of a prototype design. The heuristic sets expand that of Nielsen and Molich [576], which has been popular in GUI evaluation, and include the following heuristics: Visibility of system status; Match between system and the real world; User control and freedom; Consistency and standards; Error prevention; Help and guidance; Flexibility and efficiency of use; Aesthetic, minimalist and engaging design; Help users recognize, diagnose and recover from errors; Context preservation; and Trustworthiness.

2. In evaluation, they are used to assessing usability against a baseline design or assess how ready a design is for deployment; this may be formative or summative evaluations. For example, a cognitive walk-through can be used to assess if users face problems learning to use a new system. Information systems for nurses, for example, often need to work intuitively. In a recent study [234], five expert assessors evaluated a nursing information systems. They identified 24 unique usability problems. The authors argued that by fixing some of the most critical issues, the learning time and cognitive load experienced by nurses could be improved.

3. In accident investigation, they are used to identify factors that can increase the chance of accidents. Human error analysis, for example, is used in aviation in accident analysis [870].

Analytic methods are appealing because of their cost-efficiency: The savings can be remarkable compared to an empirical study. Unlike other forms of evaluation (such as think-aloud studies, see Chapter 42), analytic evaluations are relatively quick to perform. The cost of an analytical evaluation is primarily composed of the cost of the salary for the evaluators. However, not anyone will do. For high-quality evaluations, evaluators need to be trained in the methods. The higher the expertise, the better the results. Analytic methods can also be used on all types of representations of designs, from a system in use to an early mock-up of a design idea. Finally, analytic evaluation methods can help identify solutions to a problem. If a heuristic like "Help users recognize, diagnose, and recover from errors" is breached, it may trigger ideas on how to fix the situation.

However, analytical evaluation methods are not a replacement for empirical studies. Compared to empirical studies, analytic methods tend to have high rates of false positives and false negatives. Cockton and Woolrych [160] found that 65% of the predictions from a heuristic evaluation were incorrect; Hvannberg et al. [361] found that 62% of the problems could not be found in a usability test.

Guideline	Cognitive Factor	Example
Design for a biased perception	Perception is biased by experience and goals	Placing 'OK' button to an unfamiliar place may make users not see it
Design for figure/ground perception	Perception evolved to see structure in environment	Structure information into wholes using visual cues like proximity, closure, and common area
Design for easy scanning of text	A visual hierarchy (e.g., headings, bulleted lists, tables) helps form structure	Instead of writing out a paragraph of text, show it as a heading and bulleted list of steps or points
Design for limited memory	Human working memory is limited and easily distracted	Avoid moded designs, i.e. that effects of input depend on which state the interface is in, as the mode can be forgotten by user
Design for goal-driven attention	Attention is biased toward goal-relevant items in an interface	Present task-relevant information in a way and position that is relevant for the on-going task of the user
Design for recognition, avoid insisting on recall	Recognizing stimuli is faster and less error prone than actively recalling facts	Show options from which users can choose (e.g,. icons, images) rather than requiring them to recall names (e.g,. login names, passwords)
Design for goal-driven attention	Attention is biased toward goal-relevant items in an interface	Present task-relevant information in a way and position that is relevant for the on-going task of the user
Design for the right level of skill	Frequency, regularity, and type of practice affect the level of skill that can be expected from user	Use consistent terminology and labels throughout the user interface
Design for slips	A wrong routine can be incorrectly executed if the visual environment bears similarity with the right one	If a rare interaction has risks involved, make it visually dissimilar from routine ones

Table 41.1.: Practical design guidelines rooted on study of human cognition. Adapted from Johnson [383].

A common cause for the low hit rate is that their successful application heavily depends on the skill of the evaluator. If two evaluators were to perform the same analysis for the same case, their results might be different. In other words, *inter-evaluator agreement* tends to be low, leaving also analytic evaluation methods threatened by the evaluator effect (see Chapter 42). The limited scope of analytic methods is another threat to validity. Analytic methods pertain to a few predefined aspects of usability. Analytic evaluation methods do not reliably capture phenomena within their scope; and they never capture those outside of their scope. For these reasons, analytical methods are not a replacement for empirical evaluations. They are best seen as complementary. When correctly applied, analytic methods decrease the cost of iterative design and increase its chances of success.

In the rest of this chapter, we look at five different approaches to analytic evaluation.

1. Heuristic evaluations, which capture theories and practitioners' experiences in what causes problems in interaction;
2. Human error identification method, which is used to identify the potential for human error;
3. Cognitive walkthrough, which builds on a theory of human cognition to propose how to evaluate the learnability of a design;
4. Keystroke-level modeling, which is a simple mathematical model of experienced users' performance that predicts users' task completion time; and
5. Automated usability evaluation, where a yardstick for user interfaces is applied automatically to interactive systems.

Paper Example 41.0.1 : Analytic evaluation for inclusiveness

Analytic methods are cost-efficient methods often considered to be limited to usability. But plenty of methods have been developed for other factors that is important in HCI, including inclusiveness. How could one evaluate if a design is inclusive?

GenderMag is short for Gender Inclusiveness Magnifier [111], an evaluation method created to investigate the inclusiveness of an interactive system. The idea is to combine personas (see Chapter 15) with walkthroughs, which we talk about later in this chapter.

The method presents three customizable personas: Abi, Patricia/Patrick, and Tim. While Abi and Tim represent deliberately gendered personalities, Patricia/Patrick is decidedly neutral in those aspects and stronger in other facets (e.g., learning style). All three have five facets describing them in more detail: motivation, computing self-efficacy, risk attitude, information processing style, and learning. Self-efficacy here refers to beliefs about one's ability to complete computing tasks successfully. The evaluator starts by filling in these characteristics in light of user research data. They also walk through like actions and related those to the personas. In all steps, the evaluator can identify problems with the interactive systems.

An evaluative study reported a high positive rate in predicting a software's inclusiveness.

InclusiveMag is a generalization of GenderMag that considers not only gender but eight diversity dimensions [522].

41.1. Heuristic Evaluation

The term *heuristic* refers to a 'rule of thumb'. It is a rule of sort, but typically loosely defined. For example, if you love cooking, you may know a heuristic for checking whether an egg is edible or not. If you place an egg in boiling water, and it floats, instead of sinking to the bottom, it is likely to be inedible. In computer science, heuristics refer to rules to solve computational problems. For example, a useful heuristic in computer science is the following: a list of numbers can be sorted in increasing order by following a rule: *Pick any item: if it is smaller than the one before it, swap the two.* If you continue applying this rule, eventually the whole list will be sorted. In HCI, heuristics refer to the best practices identified by practitioners working in the field. They are typically expressed as 'dos and don'ts'. Applied to a design, they can identify potential problems in usability.

A usability evaluation heuristic is a rule for evaluating a user interface. They are used to detect probable *usability problems*. In a heuristic analysis of an interface, an *evaluator* is provided *a set of heuristics* and the interface. The task is used to detect *breaches* of the heuristics; in this case, the heuristics serve as the yardstick for the evaluation. In case of breaches, the inference is that users will experience usability problems in using the system or that the usability of the system will be negatively affected.

Numerous heuristics have been presented for different platforms and uses, from graphical user interfaces over games to web pages [690]. The most popular heuristic evaluation

method is attributed to work by Molich and Nielsen in the 1990s [540, 576]. They clustered hundreds of evaluation guidelines available at that time. They found that a small set of guidelines described much ground. Based on feedback and a factor analysis study, a revised list, still considered the main heuristic set.

Visibility of system status The current state of the system should be visible to the user. A simple example is a progress bar that indicates the progress of a long-term operation (such as downloading a large file).

Match between system and the real world The user interface should follow the language and any relevant conventions users are already aware of. If a user has to look up a term to understand it, then usability is reduced.

User control and freedom Users should be encouraged to explore different ways of achieving their goals in the user interface. To allow this, it is important that users can reverse their actions. The ubiquitous *undo* and *Redo* functions are examples of interface features introduced to support this heuristic.

Consistency and standards First, user interfaces should follow standard platform, system, and industry conventions. This is sometimes known as maintaining external consistency. Second, similar interface features should be consistently labeled and visualized throughout the application or system. This is known as internal consistency.

Error prevention The user interface should be designed to prevent errors, for example, by displaying a warning and requiring user confirmation before a non-reversible action is triggered, such as deleting a file.

Recognition rather than recall It is more difficult for users to recall from memory how to trigger an action than it is to recognize a mechanism for triggering the action shown on the display. Therefore, any information required to trigger common actions, such as labels, buttons, and menu items, should be either immediately visible or easily retrievable.

Flexibility and efficiency of use Since users inevitably vary in their proficiency of a user interface, it is often effective to provide interface features that tailor to different users. A simple example is providing keyboard shortcuts for menu items and toolbar buttons that allow an expert user immediate access to these functions, while a novice user can still use direct manipulation to easily locate them (albeit at a slower pace). Another strategy is to allow users to customize the user interface or have the user interface adapt to the user.

Aesthetic and minimalist design Ensure that the user interface focuses on content and information essential for allowing users to achieve their primary goals. Avoid providing information that is rarely relevant and avoid introducing user interface elements that may distract users.

Help users recognize, diagnose, and recover from errors Provide error messages that are understandable by users and offer a clear solution path to rectify the problem.

Help and documentation If documentation is required, ensure that it is focused on aiding users in their tasks and is easy to search. If possible, present documentation within the context it is required, such as a step requiring the user making a decision. Any help is best provided by listing the concrete steps necessary for the user to carry out the task.

41.1.1. How to do a Heuristic Evaluation?

Heuristic evaluation is best learned through an exercise. Take your laptop and open the settings panel of your OS (operating system). Decide a setting you want to change, such as the way your mouse cursor behaves (its 'transfer function'), and the particular value you want to set it to. Now go to starting view and execute the required (i.e., correct) actions and write down all steps on paper. Then go through each step and compare against each heuristic of Molich and Nielsen given in text. What to do with the result? First, count the number of heuristic violations. Is this an acceptable number? However, counting does not tell much about the real-world relevance of those violations. Second, think about your users: what is important to them and which violations would be harmful to them? Now classify the violations into three classes of severity: (1) minor, (2) critical, (3) catastrophic. Mark the severity of each violation in the list above. If you have a single critical violation, you can consider that the design failed.

Slightly more formally, the resources that evaluators use in a heuristic evaluation are (a) the choice of heuristics and associated training materials, (b) a way of going through parts of the interactive system that is being evaluated, (c) a process for evaluation, and (d) analysis and reporting of the problems identified.

Choice of Heuristics

For choice of heuristics, the most common is the Molich and Nielsen heuristics described above. This is a safe option. Typically, each of these heuristics is associated with a more extensive explanation or teaching material. For example, the visibility of the system status concerns whether the user knows what an interactive system is currently doing and what it is possible to do with it. For instance, showing that the CAPS lock key on a keyboard is active (for instance, by a light on it or a "CAPS LOCK ON" text field on a display is a way of making the system status visible. Material for the above heuristics is available in Nielsen [574].

However, heuristics that are *customized* for a particular use situation or user group can have better hit rate. But how to construct an effective heuristic set? Some heuristic sets are based on an authorative expert's view: 'This works but this doesn't'. This is fair: to the extent that these are empirically validated and their scope is well known, it does not matter where heuristics come from. However, to develop a set of heuristics that better captures consensus in a specific application area, several experts must be involved. This can be done either pre-hoc, by including several experts to jointly develop

heuristics, or post-hoc, by clustering heuristics that experts have already developed. Such specialized examples include heuristics for evaluating displays that are distributed in the environment [502], mobile computing [68], and the playability of games [186]. A longer worked example of heuristics developed for interactive AI given in the Paper Example box below.

Choice of System Parts to Evaluate

The original instructions for heuristic evaluation were unclear as to which parts of a system to evaluate and how to go through them [161]. Several different approaches have emerged. The evaluator may follow a system scanning approach. In that, the evaluator scans the system, looking for a breach of heuristics. This helps cover most of the features of a system. Alternatively, the evaluator may use a set of tasks that are representative of what users would like to do with the interactive system. This helps to keep the evaluation focused on what users would want to do and helps to uncover dependencies and inconsistencies in a larger task. The choice is up to the evaluator, but the task approach is often preferred because it helps keep false positives somewhat better in check.

Which heuristic to consider and when? Some recommendations ask us to focus on one heuristic in one turn. This makes it easy to scan the system or go through a task keeping just one heuristic in mind. An alternative is to consider all heuristics at each step of a task or for each feature. Again, the choice is up to the evaluator: The former runs a risk of false positives (e.g., because an evaluator might report a breach of a heuristic because they focus on that heuristic) but the latter is harder for the evaluator (i.e., it is easier to forget particular heuristics).

Process of Evaluation

A heuristic evaluation is typically done alone. However, it is desirable to use several evaluators who first work individually and then combine their results. One reason for this is attention to reliability: If individual evaluators might not spot the same problems, it is useful to combine the work of several evaluators. This is a general approach to improving reliability—it is for the same reason that you may ask about the same construct with multiple questions in questionnaires, see Chapter 13). Moreover, working together, they may bias each other to see the same usability problems. For these reasons, the process is typically that heuristic evaluators first perform individual walkthroughs and then discuss their results.

Paper Example 41.1.1 : Heuristic evaluation of interactive AI

Heuristic sets can be developed for a given class of technologies. As discussed in the text, this can be done through expert panels and empirical studies. A timely example is heuristics for evaluating systems that use interactive AI. Amershi et al. [20] proposed and developed a list of 18 heuristics:

1. Make clear what the AI system can do
2. Make clear how well it can do what it does
3. Time services based on context
4. Show contextually relevant information
5. Match relevant social norms
6. Mitigate social biases
7. Support efficient invocation
8. Support efficient dismissal
9. Support efficient correction
10. Scope services when in doubt
11. Make clear why system did what it did
12. Remember recent interactions
13. Learn from user behavior
14. Update and adapt cautiously
15. Encourage granular feedback
16. Convey the consequences of user actions
17. Provide global controls
18. Notify users about changes

An example is given in Figure 41.1.

Analysis and Reporting of Usability Problems

Based on the heuristics and the choice of system, an evaluator will have made a list of notes on the problems.

Analysis is often under-appreciated in usability evaluation [247]. However, because

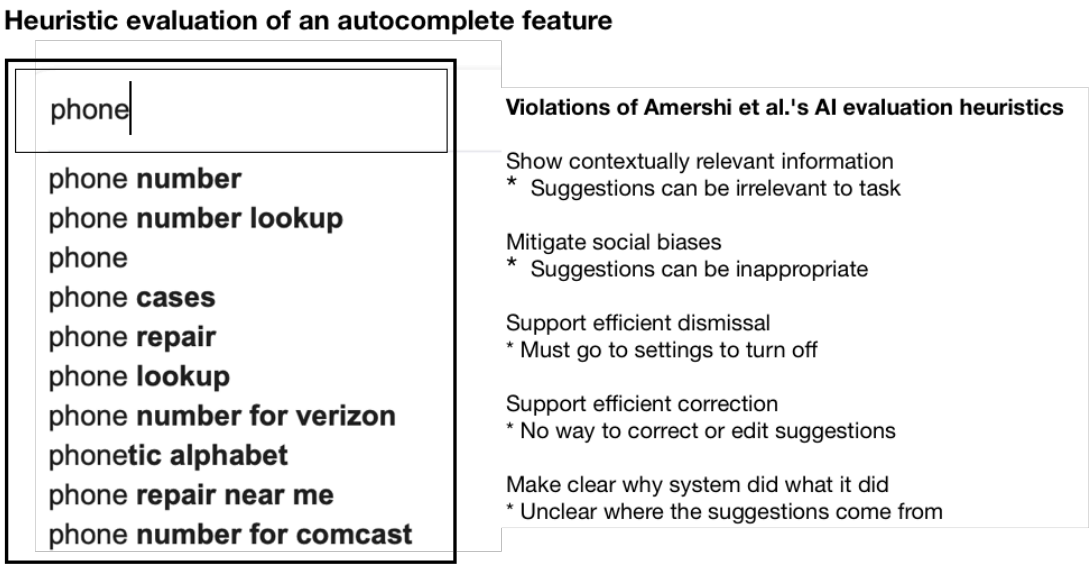


Figure 41.1.: Example of heuristic evaluation applied to an autocomplete feature in text entry. Based on Amershi et al. [20].

heuristics are just resources that an evaluator judiciously applies, considering and analyzing the potential problems is an important step. Does it seem plausible that a particular potential problem is *actually* a problem? What is the cause of a particular problem? What could be done, if anything, to resolve a problem? A couple of resources can help in the analysis of problems.

- Frequency. It is useful for an evaluator to assess how often a problem would occur for prospective users. One commonly used scale separates (a) rare, (b) occasionally, and (c) frequent.
- Severity rating. This expresses, according to the evaluator, how serious a problem is for users. One commonly used rating scale is (1) Minor: the user is temporarily delayed. (2) Serious: the user is delayed significantly but can eventually complete the task. (3) Catastrophic: prevents the user from completing a task.
- Persistence. This assessment concerns whether the problem is a one-off occurrence that users will learn to work around, or something that will bother them time and again.
- Cause. What is the cause or reason for the problem? Even if an issue is pointed out by a heuristic, it may not be clear what in the interactive system that is behind a problem; this is even a bigger problem in empirical evaluation (see Chapter 42) where mapping users' expressed frustration to issues in the system may be very difficult.

Description	Difficulty for user	Cause	Frequency	Severity	Solution
"Open report" is hidden	The user needs to search the entire page to find the link	The link is located in a spot that the user might not notice.	Frequent: The user will experience this difficulty every time they log in	Serious: the user may struggle to find the link for a while	Move the link "open report" to the upper-left corner

Table 41.2.: Example of a usability problem reporting format.

- Redesign suggestion. This asks the evaluator to consider how the interactive system might be redesigned to avoid the problem. Redesigns can help evaluators avoid reporting problems that have no solution; it also impacts the clarity and usefulness of the problem to other people [347].

The reporting of problems has been shown to be an important resource in the analysis of problems. The above items, for instance, can be simultaneously addressed using a structured reporting format, such as that shown in Table 41.2.

41.1.2. The reliability of heuristic evaluation

Heuristic evaluation has a role in quick assessment of designs, however, is not a panacea and not a replacement for empirical studies. The known benefits and drawbacks of heuristic evaluation include:

- High efficiency: Some usability problems can be spotted with little effort and experience
- Limited scope: Heuristics are limited to aspects of usability that can be attributed to visible parts of the UI.
- High false positive and false negative rates: Many important problems are not found, and some problems are identified that are not problems.
- No guarantees: The results cannot be trusted to be comprehensive, reliable (large variability), nor generalizability (except for most obvious usability problems)
- High variability: Evaluators, even experienced, show drastically varying hit rates.

One core reason for these issues is that evaluators, even experts, are imperfect as 'signal detectors'. Every time an evaluator inspects some facet of a design, with some probability, some problem is found. However, this can be a false positive or false negative error. In

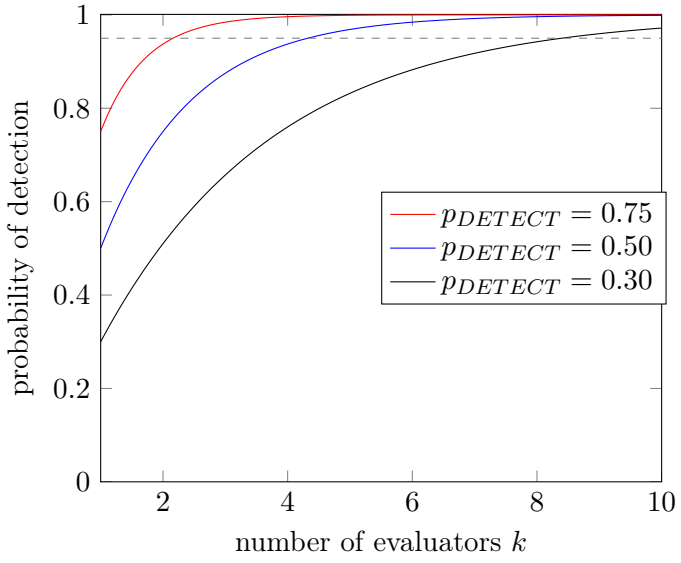
other words, the evaluator may have falsely 'found' an error while in fact there is one, or missed an error that actually is there, believing that there is none. The more complex the system, and the more problems, even obvious ones, the more the evaluators will miss. A statistical analysis illuminates this.

From a statistics perspective, an attempt to use a heuristic to detect a violation can be thought of as *an experiment*. It is a trial with two possible outcomes: success or failure. You can think of it as analogous to the tossing of a *biased coin*: either a usability problem is detected (success) or not (fail). In statistics, experiments with two outcomes are called Bernoulli trials. We use Bernoulli trials here to answer a central question in heuristic evaluation: What is the proportion of problems that we expect k evaluators to be able to find? Given n possible usability problems,

Number of evaluators

Bernoulli trials provide insight into the question of how many evaluators are needed. We start with the case of a single usability problem and ask how k (the number of evaluators) affects the probability of its detection. Let us assume that our k evaluators are often imperfect. They can spot a real usability problem with probability $0 \leq p_{DETECT} \leq 1$. We further assume that *if* an evaluator detects a problem, that it exists. In other words, for simplicity, there are no false positives in this model.

Now, let the random variable $\mathbf{X}$ denote the number of evaluators needed such at that *at least one* detects the usability problem. This follows the Bernoulli distribution, $\mathbf{X} \sim B(p)$. In a Bernoulli process, a Bernoulli trial is repeated for k times. In this case we are interested in the occurrence of *any* true positive within that sequence, which is given as $p(\mathbf{X} > 0) = 1.0 - p^k_{DETECT}$:



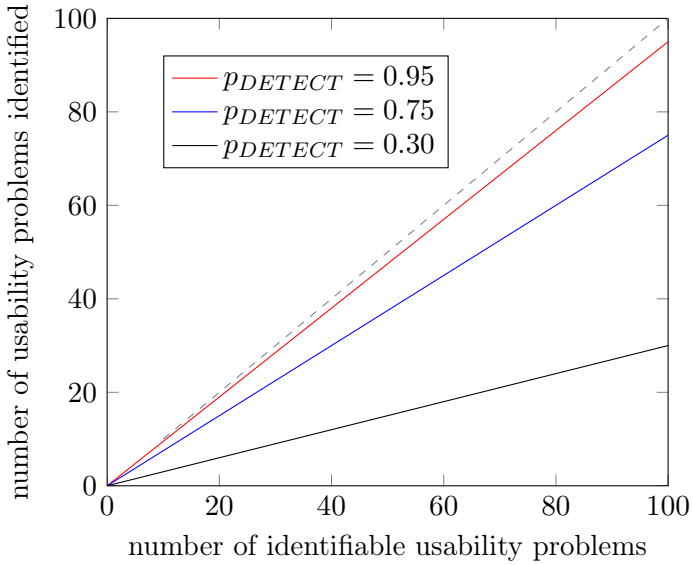
The plot shows that we only need two expert evaluators (red line) to detect a usability problem with high probability (≥ 0.95), but four or more when they are intermediately skilled (blue). Note that for a novice, for example, a starting computer science student,

detection probability is between 7 and 75 % [576], and often around 30%. The black line shows that seven or more such computer science students would be required for reliable detection. In summary, the skill of the evaluator matters.

Coverage of problems

We can now look at the case where we have several usability problems. Let us assume that, unbeknown to us, a system has n usability problems. We ask a group of equally skilled evaluators to evaluate it,assuming that their individual detection probability to be p_{DETECT}.

Let the random variable $\mathbf{X}$ denote the number of usability problems observed by the evaluators together. Now the number of problems found in n trials with detection probability p_{DETECT} follows a *Binomial distribution*: $\mathbf{X} \sim B(n, p_{DETECT})$. Now, the number of usability problems found is simply np_{DETECT}:



Putting the first and second results together, we learn the following.

1. High coverage of real problems is hard to achieve without experts (red). Only experts provide a high detection probability.

2. Evaluations by a single evaluator are inherently unreliable. Even an expert evaluator will miss an unacceptably large proportion of 'obvious usability problems' (blue)

3. On the other hand, even a poor evaluator will find *some* usability problems.

The moral of these analyzes is to show that heuristic evaluation is not a silver bullet nor replacement for empirical evaluations. However, well-trained experts can detect a nontrivial proportion of true problems. This calls for caution in applying heuristic methods and rigor in training and evaluating their accuracy.

41.2. Identification of human error potential

Human error identification (HEI) is a class of analytic evaluation methods for identifying the possibilities of human error in interaction. HEI techniques were originally developed for safety-critical applications, but the main idea generalizes to most state-based user interfaces. It can be used to spot interactions where users might be confused, or make a mistake or error.

One key difference from heuristic evaluation is that HEI is preceded by task analysis (Chapter 15). In practice, this means that the goals and states of an interface are first identified. A user is then 'simulated' going through the states toward a goal, while asking what kinds of errors might happen.

Consider a vending machine and the task of buying a can of soda. This task is actually quite complex. It consists of several steps, more than one might think, such as finding the desired soda on the list, pressing the button, finding wallet, finding credit card, inserting it, entering PIN, waiting for the soda, and picking it up. The basic idea of HEI is to look at *transitions* between steps. What can go wrong when moving from one step to another? For example, why would a user not find the credit card or insert it to the wrong place? HEI can reveal possibilities for taking wrong steps.

A central concept in HEI is *state*. At any given time, the interface can be in one state where it is ready to receive input from the user. Some actions are available, but others not. The system communicates its state and possible next actions on its display, for example via text, graphics, etc.

To utilize HEI, we should first enumerate all states of the UI. We then form a matrix showing possible transitions between states. We then label each sell to show if a transition is available: (1) legal, (-1) illegal (erroneous; should not be done by user), and (0) not available.

This state matrix is now used as a basis of simulating users. Take a user persona and a particular task of that persona. Then start from the state in which the user would encounter the device. Then enumerate possible reasons why the user might take *wrong action* in that state.

HEI provides three main categories to this end, but in principle any cause of error can be used:

1. User confuses the correct action with something else
2. User confuses the state of the machine
3. User selects erroneously

For each illegal (wrong) action, assess the consequences of that error. What would happen and how severe would the cost be?

Can HEI techniques actually predict errors? To answer this question, Baber and Stanton observed public vending machine use for 24 hours, totaling 300 observations of transactions [32]. They categorized and tabulated the found usability problems, finding problems such as money insertion (27%), zone or destination selection (22%), ticket type or

travelcard selection (21%) are the most common. This formed their ground-truth dataset. Independently of these observations, they carried out HEI. Their task analysis diagram had 11 subtasks with some substructures at depth of there, reflecting the non-triviality of the machine. Error analysis was performed by taking a state and enumerating all available actions. The transitions caused by these actions were then mapped to other states, forming a matrix. The matrix marks legal (1), illegal (-1) or not available (0) transitions.

Their analysis exposed the following problems with a vending machine, which may be familiar to many of us:

1. Not understanding what to do
2. Selecting wrong ticket
3. Selecting wrong station
4. Selecting wrong zone
5. Problems in inserting money
6. Confusing mode
7. Pressing wrong buttons
8. Using a machine that is closed
9. Confusing return from cancel
10. Use a machine that is waiting

Comparing these with the ground truth dataset, the authors concluded that one evaluator using the method could identify more than 80% of real usability problems. This analysis took about 3.5 hours per evaluator, which is a significant saving over real-world observations.

41.3. Cognitive walkthrough

Cognitive walkthrough is an analytical evaluation method based on mental simulation of the way users think. It is an instance of a broader class of *walkthrough methods* used across engineering disciplines, for example, architectural walkthroughs in architecture and code walkthroughs in software engineering.

In cognitive walkthrough, similarly, an artifact is inspected systematically, in a step-by-step manner, and evaluated against criteria. In this sense, it is similar to HEI. What makes the cognitive walk-through special is that evaluation criteria are related to thinking and cognition. The method relies on the evaluator *simulating in his/her mind* to determine whether a user *might* succeeds or fails in the interaction. The user is simulated in guessing and exploring how to use an interface. However, walkthroughs are not just any form of imagery, but follow systematic procedures and conceptual apparatuses.

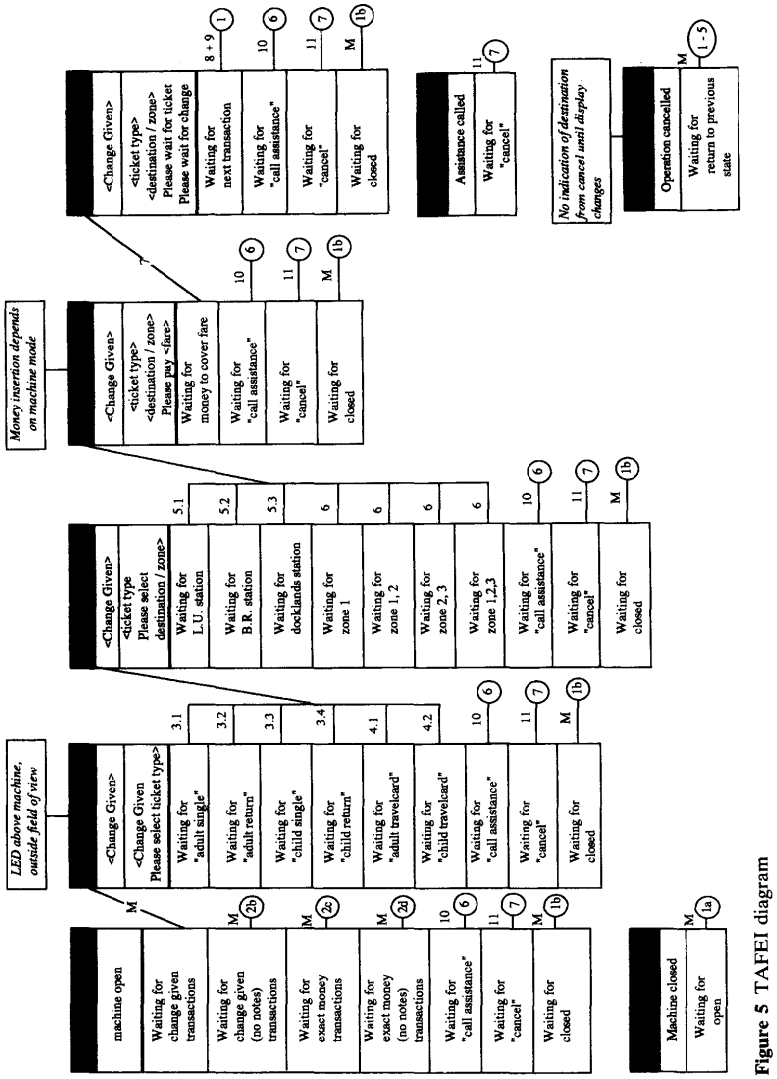


Figure 41.2.: Human error identification methods focus on the identification of possibilities for transitioning to wrong states. Here, a diagram showing the possibilities of taking a wrong step in vending machine operation [32].

The goal of cognitive walkthrough is to expose possible problems impairing the *ease-of-use and learnability* of a system. The method is recommended for understanding how novice users may figure out how to use a system. Its scope is different from other methods discussed in this chapter, which focus on user performance, errors, and usability problems.

Cognitive walkthrough as a method is straightforward but substantially more laborious than heuristic evaluation. The inputs to the method are (1) the user interface, (2) a task scenario that tells what the users are supposed to accomplish, (3) assumptions about the users and the contexts of use, and (4) a sequence of actions that complete the tasks. Task

analysis is needed to prepare point (4). In most cases, the analysis of subtask *sequences* suffices.

There is good evidence that it can predict a significant part of learnability-related problems. In the original comparative study by Lewis et al. [462], cognitive walkthrough detected 50% of problems exposed in an empirical user study. Similar results have been observed in later studies.

41.3.1. How to do a Cognitive Walkthrough?

In a walk-through session, tasks are demonstrated to a team in a step-by-step manner, attempting to explain, plausibly, how the user might solve four issues related to use:

1. Will the user try to achieve the right effect?
2. Will the user notice that the availability of the correct?
3. Will the user associate the correct action with the intended effect?
4. If the correct action is performed, will the user be aware that the task is progressing as intended?

If a plausible explanation cannot be given, this is recorded as a critical issue. These critical issues are reviewed together with the team to identify design gaps and set goals for the next iteration.

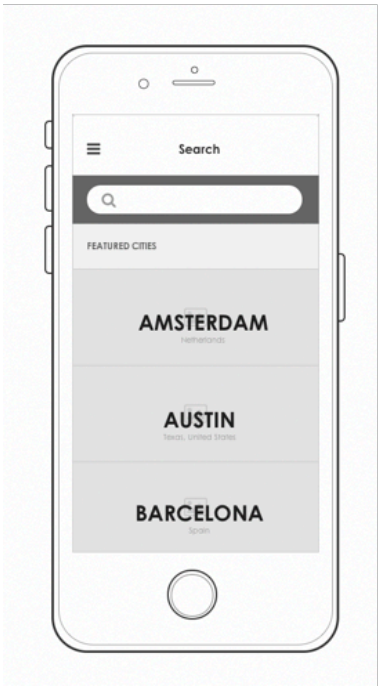
41.3.2. A theory of how people learn via exploration

Cognitive walkthrough is rooted in a theory of how people learn interfaces. Informally it could be called a theory of how people *guess* what to do next. The theory of *cognitive exploration* was proposed by Polson and Lewis [653].

The crux of the theory is this: To complete a task, the user must set a goal relevant to the task and achieve the subgoals on the way there. However, each subgoal consists of actions. Each action, finally, insists that the user *selects* the right action, *executes* it correctly, and can *confirm* that it is indeed progressing the interaction as desired.

How do people solve this then? They must be able to cross two 'gulfs': *the gulf of evaluation*, or determining the right action for the goal, and *the gulf of execution*, or successfully executing the action and confirming that it was a success (see Chapter 18). For an artificial agent learning, this would be a hard task, because it requires generalizing from past experiences to a novel, previously unseen task.

According to Polson and Lewis, people first establish *goal structure*. This goal structure associates sub-goals to the top-level goal. For example, when cooking pasta, the subgoals are related to acquisition of ingredients, preparation of the cooking space, familiarization with the recipe, etcetera. However, the goal structure may be incomplete. The missing subgoals must be figured out on the go, which puts further emphasis on how well the user interface guides the user.



Task: Find hotels in Helsinki

Walkthrough questions

1. Will the user try to achieve the right effect?
2. Will the user notice the availability of the correct action?
3. Will the user associate the correct action with the intended effect?
4. If the correct action is carried out, will the user be aware that the task is progressing as intended?

Report

1. Yes, assuming familiarity with the concept of a search box
2. Yes, assuming that the icon is large enough
3. Yes, assuming familiarity with search boxes
4. Yes, the changing of the display will be clear

Figure 41.3.: Cognitive walkthrough: Simulating a user solving of a task step-by-step.

A key part of the theory concerns the representation of a subgoal. A subgoal is a representation associated to (1) other subgoals, (2) actions, and (3) perceptual cues. Consider the sub-goal of turning on a stove as an example. This subgoal is associated to the top-level goal of making tea, as well as the previous and following subgoals (e.g., putting the kettle on the stove). It is associated with actions that are needed to accomplish the goal, such as turning a knob. It is also associated with perceptions, or cues, such as those of the knob and the stove and the kettle. According to the theory, these cues activate the representation of the associated goals and actions. Just seeing the knob activates the subgoal of turning on the stove. However, when the subgoal is completed, the representation is deactivated. Thus a representation can be *activated* and *deactivated* (inhibited) by its associates. When interaction flows well, percepts activate the right subgoals, which activate the right actions. When taking that action, the state changes such that the next subgoal is activated again, and so and so forth. However, when there is no highly activated action, the user must explore, just try out something. This inevitably leads to errors. On the other hand, the erroneous paths will be deactivated, and a better goal structure learned over time.

One of the benefits of the theory is that some of the issues can be addressed in design prior to evaluation. Lewis et al. [462] propose four design tactics for improving guessability:

1. Make the repertory of available actions salient.

2. Provide an obvious way to undo actions.
3. Offer few alternatives.
4. Require as few choices as possible.

CW is more time-consuming to apply than heuristic evaluation, and it insists on a trained evaluator. Deployment in software teams has been reported to face problems due to time pressure [767]. Similarly to heuristic evaluation, the high rate of false negatives in cognitive walkthrough is a recognized problem. While capturing some problems is always better than capturing none of them, one must remember that a large proportion of problems is *not* identified. While cognitive walkthrough complements heuristic evaluation, neither is comprehensive. Cognitive walkthroughs are best used in early, noncritical stages and should preferably be carried out by experienced evaluators.

Several adaptations of the method have been created for various purposes. Mahatody et al. [496] provides an overview of variants and extensions of CW. A faster version has only two questions [767]:

1. Will the users know what to do at this step?
2. If the do take the right action, will they know that they did the right thing and are making progress towards their goal?

41.4. Keystroke-level modeling

Keystroke Level Model (KLM) is a simple mathematical model to assess the performance of tasks. In particular, it can predict *task completion time* of experienced users. KLM was introduced by Card, Moran, Newell in their 1983 book *Psychology of Human-Computer Interaction* as a simplified version of GOMS [129], a more comprehensive model based on a cognitive architecture (Chapter 5).

KLM deals with tasks that have sequential sub-structures, such as form-filling, text editing, data entry, or setting manipulation. KLM is limited to sequential tasks with clear task boundaries, no parallelisms (e.g., multitasking), and little dependencies between subtasks. By assuming that performance in each sub-task is independent of performance in the others, task completion time can be modeled as a sum of time spent in the sub-tasks.

KLM is ”keystroke-level” in the sense that higher order control nor carry over is not assumed. In other words, behavior is deterministic, goes form one subtask to another, and task completion time is simply the sum of time spent in ”atomic” responses and actions. There is no learning or memory assumed in the user. Every atomic response takes place regardless of what happened in the past. Although KLM was derived from GOMS, it is not a cognitive model but a performance evaluation model. It does not say anything deep about how the mind works in interaction. However, even if the model is simple, it serves well user interfaces that are sequential in nature.

41.4.1. How to do a KLM analysis?

KLM starts by analyzing user's task to find the most likely way, or ways, of accomplishing it. This task performance is broken down according to three categories of *operations*: physical, mental, and system. Mental operations refer to events such as recalling a command name or verifying that an answer is correct. The system operation is the system response time, the time spent waiting. These operations are counted, and the times spent in each are estimated using guidelines and look-up tables. Task completion time (T) is then their linear sum:

$$\begin{aligned} T &= t_K && \textit{keystroking} \\ &+ t_P && \textit{pointing} \\ &+ t_H && \textit{homing} \\ &+ t_D && \textit{drawing} \\ &+ t_M && \textit{mental operation} \\ &+ t_R && \textit{system response} \end{aligned} \tag{41.1}$$

But how to determine these values? These have been empirically estimated [127]. *Keystroking* is around .12 - 1.2 s, .28 for most users. However, the level of expertise affects this. An expert typist has an average keystroke time of .12 seconds, whereas someone with less exposure to keyboards has 1.2 s. *Clicking* a mouse button is 0.20 s. *Pointing* averages at 1.1 s, but with varying movement distances, should be determined by a Fitts' law model. *Homing* is the act of moving the hands between the mouse and the keyboard. It takes about 0.4s. *Mental operations* are in the range of .6 - 1.35 s, but 1.2 s is recommended. Mental operations are required when initiating a task, making a strategy decision, recalling something from memory (e.g., password), finding something on the screen, thinking about what to do, or verifying the correctness of inputs. *System response time* must be measured and depends on the per case.

The procedure for KLM modeling includes eight steps [129]:

1. Choose one or many representative task scenarios.
2. Specify the design to the point that keystroke-level actions can be listed for the scenarios.
3. For each scenario, identify the most likely ways users will accomplish the tasks.
4. List the keystroke-level actions and the physical operators involved in doing the task.
5. If necessary, include wait operators for those those time when the user is waiting the system to respond.
6. Insert mental operators for when user has to stop and think.
7. Look up the standard execution time to each operator.

User action	KLM event	time (seconds)
Reach for mouse	H:mouse	0.40
Search "Replace"	M:visual search	1.20
Move pointer to "Replace" button	P: menu item	1.10
Click on "Replace" command	K:click	0.20
Home on keyboard	H:keyboard	0.40
Type word to be replaced "might"	K:type	0.60
Reach for mouse	H:mouse	0.40
Point to correct field	P:field	1.10
Click on field	K:click	0.20
Home on keyboard	H:keyboard	0.40
Recall word	M:recall	1.20
Type new word "will"	K:type	0.48
Reach for mouse	H:mouse	0.40
Move pointer on "Replace All"	P:button	1.10
Click on field	K:click	0.20
Wait	R:system	0.50
Verify	M:verify	1.20
	Total	11.08

Table 41.3.: Example KLM model in a replace task in text editing.

8. Sum up execution times for the operators. This is the estimated task completion time.

41.4.2. A Worked Example

Let us consider a simple example with physical, mental, and system operations. An expert user is editing a document and wants to replace all words "might" with the word "will". Table 41.3 shows the breakdown of actions. They can be summarized as follows.

$$\begin{aligned} T &= t_K + t_P + t_H + t_M + t_R \\ &= (0.60s + 0.48s + 3 \times 0.20s) + (3 \times 1.10s) + \\ &\quad (5 \times 0.40s) + (3 \times 1.20s) + 0.50s \\ &= 11.08s \end{aligned}$$

The analysis reveals that the user spends almost one-quarter of the time just moving the hand between the mouse and the keyboard.

41.4.3. Limits of KLM

In empirical studies, KML has proven to be effective at predicting task completion time within a large but acceptable limit of tolerance. In a recent evaluation, KLM's predictions were compared against 20 expert users [390]. Across six tasks and UIs, KLM predicted human performance in four tasks. Average difference to human data was 5.5 s (range: 0.8 - 13.35) and slightly smaller for GOMS.

While the time spent in system operations can be exactly measured and the time in physical operations can be estimated from observational data, mental operations pose a problem. How to identify them? Best practices in KLM modeling suggest that it is more important to get the type and number of operations right than to determine the order in which they occur. Moreover, very different interfaces paradigms should not be compared without identifying better values for M. For instance, command-language interfaces and GUIs have different requirements for recall, so using the single constant for both will bias the estimates.

There is a number of other criticisms levelled at KLM, including the following [602]. First, KLM assumes expert performance, a user who is able to directly use the most efficient strategy and carries out tasks without error. In this respect it complements cognitive walkthrough that focuses on how people 'guess' what to do. Second, KLM ignores flexibility in human activity. In Chapter 21, we learned about behavioral strategies and how they adapt to the structure of the task. Full-fledged cognitive models are needed when behavioral adaptation is a defining part of users' performance. Third, KLM ignores the variation in performance. It is known that there are dramatic inter- and intra-individual differences even in the lowest levels of actions, such as in button-pressing. However, KLM collapses that variability to a single-point estimate. If the statistical distributions of KLM operators were known, they could be used to sample value combinations. There are attempts to collect such distributions among different factors, such as age [274]. Fourth, KLM assumes no parallel activities in the mind; even though dual- and multitasking is a pervasive aspect of computer use.

41.5. Automated Usability Evaluation

In recent decades, much research has been done in trying to eliminate the need for an evaluator and make usability evaluation method automatic [370]. The core idea of automated usability evaluation is that an evaluation tool encapsulate some yardstick for good interaction. That yardstick may be that an interactive system is accessible, that it is easy to perceive the information on the display, that links are not broken, and so on.

Let us consider some examples. Aalto Interface Metrics [611] takes a web page as input and computes its accessibility, aesthetics, and support for accurate color perception, among others. The evaluator of a web page may use these descriptors to make inferences about how well the web page supports users. For accessibility, the world wide web organization maintains a list of tools that help automatically check (see https://www.w3.org/WAI/). These tools typically takes a web address as input and creates a list of possible or likely issues with accessibility. However, their scope is more limited than that offered by HEI,

CW, and KLM.

41.5.1. Interactive Modeling Workbenches

Computational cognitive models simulate human cognition through the stepwise execution of a program. Generally, computational models are superior to KLM and cognitive walkthrough thanks to their ability to simulate cognitive states, strategies, and motor actions. This allows insight beyond mere task completion time, into what constitutes successful and failed task performance. Generally, these tools are best suited for modeling skilled users.

A bottleneck in the use of cognitive architecture models has been that the task procedure must be provided by the researcher. In the minimum, this involves a demonstration of a sequence of actions. Interactive modeling workbenches allow you to specify a procedure simply by demonstrating it on a UI.

CogTool is a modeling workbench that uses a version of GOMS for predicting how attention, motor control, and memory work during interaction [274, 381]. Some comparisons against empirical results suggest prediction error of about 10-15% in task completion time depending on the UI and the individual. *Distract-R* is a simulation workbench specific for multitasking in driving [705]. It uses ACT-R with a model of multitasking called threaded cognition. Evaluators can specify a secondary user interface (e.g., media player UI), some limited aspects of the driving task (e.g., speed), and user (e.g., age). Distract-R then computes task completion and lane deviation predictions and provides a video of simulated driving.

41.6. Which Analytical Evaluation Method to Use?

Analytic methods are best at identifying usability problems, but no serious project should *rely just* on them, especially for critical conclusions such as those concerning safety, product launch, inclusivity, or accessibility. Analytic methods are – nonetheless – powerful and flexible complement to empirical methods.

But which method to use when? To understand the relative merits of each method, Blandford et al. [77] conducted one of the most comprehensive comparisons of analytic methods so far. They compared eight methods in the case of a human–robot interaction task. They propose a few dimensions to consider when picking an analytical method:

Scope What kinds of usability problems should be found?

Suitability What type of interaction and user group is in question?

Reliability What is the minimum reliability that should be achieved?

External validity How important is the transfer of findings to real use?

Efficiency How much information is gained per unit of resource use? Resources here can refer to the work time of an expert evaluation, human participant etc.

Persuasiveness How to communicate the results in a way that convinces the audience?

Blandford and colleagues concluded that no single method offered superior coverage of problems, but rather each conquered its own 'niche' in the space of usability problems. Concerning analytic methods, they conclusions were as follows. First, heuristic evaluation identified a broad range of issues, but was most unreliable. The interpretation of the heuristic (e.g., what is 'consistent'?) is left to the evaluator, producing large inter-rater variability. On the positive side, the openness of the heuristics leaves room for more qualitative consideration of the causes and consequences of issues, which is useful for design. Second, cognitive walkthrough complemented other methods by exposing issues related to user misconceptions, consistent with the cognitive exploration theory. Surprisingly, it encourages the identification or more issues than strictly within its scope, perhaps similarly as in heuristic evaluation, thanks to open-endedness. Third, cognitive models, especially GOMS and thereby KLM, best support the identification of system-related problems, such as the lack of 'undo', redundant operators, or long action sequences, as well as some problems in synchronizing users' actions with that of the system. Model-based methods have a focus on timing information, for example, how long actions or tasks take to complete. This is due to focus on skilled users with little considerations of errors, learning, and behavioral variability.

Summary

- Analytic evaluation relies on expert analyst systematically going through a design and procedurally checking system responses against some criteria.
- Analytic methods have low reliability when applied outside of their scope or by inexperienced analysts.
- There are analytic methods for covering different aspects of usability from performance to errors and learnability. Walkthrough methods are being developed to assess inclusiveness of a design against persona-based criteria.
- Analytic methods are best utilized as cost-efficient complements in earlier parts of design; they are not a replacement for empirical evaluation methods.

Exercises

1. Heuristic evaluation. Pick a simple user interface (e.g., a travel planner, a homepage for a volunteer organization, a magazine app). Then conduct a heuristic evaluation individually and report the problems following the format of Table 41.2. While doing the evaluation, note down difficulties, questions, and insecurities about the method.

   Next, consider the following questions and, if possible, discus it with peers who have also done an evaluation of the same interface.

- What was hard in doing the evaluation and what was easy?
- If other people have evaluated the same interface, do your evaluations agree? Do the three most critical problems overlap? Why or why not?
- Given how you, and your peers, did the evaluation, what are the three main concerns about the validity, reliability, and impact of the evaluation?
- An open question in research on analytic evaluation methods like heuristic evaluation is whether evaluators see problems and then justify them with a heuristic or whether they use the heuristic actively to identify problems. Which description best fit your work and why?

2. Cognitive walkthrough. Do a cognitive walkthrough of the same interface. Consider the same questions as in the previous exercise.

3. KLM. Take two designs for the same task, e.g. two designs for login into an information system. Conduct KLM analysis of alternatives and compare them. What aspects that might affect task completion time are not covered

4. Designing with analytical models. Take the UI you analyzed in the previous exercise and redesign it to improve predicted TCT (task completion time).

5. Comparison. Consider again the Cognitive Dimensions Framework in Chapter 27. Is that an analytical evaluation method? Why or why not?

6. Designing with KLM. A smartphone app enables a user to achieve a task through two alternative methods. In the first method, the user must push a button on the screen, navigate to a menu with five choices, and choose the fifth choice. This brings up a Yes/No confirmation dialog box. The user must answer Yes. In the second method, the user must hold a physical button down for one second until the system plays a brief audio beep and then speak a command. This results in the system playing a different audio beep to indicate it has commenced speech recognition, and yet another different audio beep to indicate speech recognition has concluded. Assuming speech recognition successfully recognized the command, at this point the system uses audio to ask the user for a confirmation. The user has to again hold a physical button down for one second until the system plays a brief audio beep and then speak "Yes". (a) Carry out a KLM analysis to understand the time durations required to carry out the task using both methods. Propose any operators necessary and estimate their time durations. Briefly motivate the operators, their estimated time durations, and the operator sequences involved. (b) Explain the limitations of the KLM analysis carried out in (a) with reference to the following design issues: (1) novice versus expert performance; (2) the uncertainties inherent in interaction when attempting to carry out the task using the two methods. (c) There is a risk an undesired event occurs when the user attempts to carry out the task. This happen when upon confirmation the user indicates "No". Draw a fault tree to understand the possible causes of such an event. (Fault trees have been introduced in Chapter 37).

42. Think-aloud Studies

Imagine you are to evaluate a computer system and put a prospective user in front of the system with a task they want to accomplish. But then nothing observable happens. Although their eyes might move and their hand approaches the mouse, you can decipher next to nothing about what they are pondering or why they hesitate when interacting. As they finally begin to use the system, they swiftly complete their task in silence. You have learned nothing about what works or does not work with the user interface.

When we evaluate user interfaces, our job would be easy if we had access to what goes on in people's heads as they use the system. We would know what they think about the elements of the user interface, how they match their goals to those elements, how they interpret feedback, and what they feel. The focus of the present chapter is a methods—the *think-aloud study*—which aims to give some form of access to what goes on in people's minds as they use computers.

The think-aloud study consists of a few core steps; depending on the purpose of the test, these may be altered. The evaluator gives the participant a set of *instructions*, typically including specifics on how to think aloud and possible tasks or activities to engage in. The participants then *verbalize* their thinking when they interact with the system under evaluation or immediately afterwards. The evaluator *captures* the verbalization and other notable events, and then *analyze* these data to derive insights into the thinking of the participants.

Think-aloud studies are common among practitioners [233]; they are used to identifying usability problems in an application (e.g., the user gives up on a task that it is possible to solve, and the user has to find a workaround to a task). Examples of HCI applications include:

- Srinivasa Ragavan et al. [770] studied how users understand formula and cells in spreadsheet applications such as Microsoft Excel. They conducted a think-aloud study of 15 professionals who read others' spreadsheets as part of their work. They found that 40 % of time is spent in searching for additional information needed to make sense of the spreadsheet. These episodes were felt as feeling overwhelmed and the users failed often. Verbal protocols exposed a pattern that the authors call ''over-the-hood" comprehension and ''under-the-hood" comprehension. Over-the-hood comprehension is when users examine what is visible on the spreadsheet, like text, numbers, and charts. Most of this understanding took place by reading labels, which took place via systematic scanning. In under-the-hood comprehension, by contrast, the users wanted to understand a formula and needed to seek information about the involved variables. This is complex because variables can refer to other formulas and other cells in complex ways. Sometimes users needed to recreate information in a spreadsheet to understand how it works.

- Oh et al. [597] studied ''AI Mirror", a user interface that tells how aesthetic their photos are based on a deep neural network model. The mirror provides an aesthetic score between 1 and 10 based on its training data. The authors used a mixed-methods approach where think aloud protocols were used to understand thinking and experience during using the mirror. The authors found that different users understand the AI method using their group-specific expertise. Users with machine learning expertise used technical concepts like ''algorithm", ''model" and ''classification" to understand the AI. For example, they speculated about the source of the dataset trying to explain anomalies in the AI's scores. Participants with photography background, in contrast, mentioned concepts like ''light", ''composition", and ''aperture" in understanding the scores. The third group represented the general public and focused on their favorite objects, beautiful landmarks and landscapes. They did not fully grasp the AI's scores and why it did not always match their own views, and they had weaker conceptual background to attempt to provide explanations.

- Tamas et al. [795] compared residential thermostat designs in Canada, with the goal of understanding how their design affects usability, and how that in turn affects energy saving practices. They compared manual, programmable, and smart interfaces using a mixed-methods approach. It included think-aloud protocols collected during task performance. The authors concluded that smart interfaces are significantly more usable. Protocol analysis showed that almost a third of users were confused when attempting to program their programmable thermostat. The programmable thermostat, with its small user interface, is overly complex for residential users who rarely need to interact with it.

Nielsen called think-aloud as the "single most important usability engineering method" [574]. However, many uses of think-aloud studies in HCI have nothing to do with usability. For instance, they may be about understanding how users learn an application through the analysis of their verbalization. Therefore, we prefer the more general term think-aloud study. Many more thorough introductions to think-aloud studies exist, see for instance Rubin and Chisnell [697] and Dumas and Loring [213].

Next, we cover the basic steps in a think-aloud study, focusing in particular on the effect of instructions, how to analyze think-aloud data, and the pros and cons of the method.

42.1. Understanding Thought Processes

Think-aloud is grounded in the belief that it provides access to what goes on in people's heads, that is, helps understand thought processes. The key argument for this belief comes from research in psychology, notably the work by Ericsson and Simon on *verbal protocols* [225]. Verbal protocols refer to the outcome of think-aloud studies, the analysis of what users said based on audio recordings of their verbalization.

Ericsson and Simon studied think-aloud as a method used across the behavioral sciences to gain insights into though processes. They were focusing on *concurrent verbalization*, where thinking aloud occurs during task completion. However, for tasks of up to 10 seconds in duration, *retrospective verbalization* may also give valid insights into though processes. In retrospective verbalization, users are asked to tell what they thought after they complete a task.

Ericsson and Simon also distinguished three levels of verbalization.

Level 1 In Level 1 verbalization, the participants are told to concurrently report what they think about in the same form as their thoughts. This happens, for instance, if participants report verbal information that they are about to enter in a user interface.

Level 2 In Level 2 verbalization, participants may transform information that they are considering from non-verbal to verbal form. This could be describing mental imagery.

Level 3 In Level 3 verbalization, participants are required to provide explanations, filter information, or relate to information not currently held in working memory (e.g., in retrospective reports).

In Ericsson and Simon's view, only Levels 1 and 2 provide valid information about the mental processes of the user because those levels only asks about information that is currently needed in short-term memory. Level 3 verbalizations significantly changes task performance and consequently the value of their think-aloud data as accurate reflections of what goes on in their thinking. For instance, giving rationales for what you do might improve your performance on a task, even if thinking aloud should slow you down.

This view has been criticized. These critiques argue that people know much more than they may express in verbalization; that is, that think-aloud studies does not give a complete picture of mental processes. Another critique is that think-aloud studies are reactive; that is, that thinking aloud may help or hinder task performance. Nevertheless, most researchers maintain that think-aloud studies give some important information on thinking. The critiques has led to a variety of methodological elaborations we discuss later.

Think-aloud studies have been used in HCI for at least 40 years [461]. As in the broader behavioral sciences, the use has been to gain insight into what participants are thinking during interaction. For instance, Mack et al. [487] explored how users learn to use text editors (see the paper example box). One widespread use of think-aloud studies in HCI has been to help identify usability problems in interactive systems. There, participants' thinking aloud is analyzed to find evidence of errors or lack of clarity in the user interfaces.

Think-aloud studies have many strengths for understanding mental processes. The key strength of the think-aloud study is its *precision* [518]. It allows for a close analysis of the user's thought processes. It is also a relatively inexpensive form of study because it requires little more than note-taking, recording equipment, or a video link. Although think-aloud studies may be analyzed in depth, less intense analysis may give useful. Actionable insights into mental processes or may help discover usability problems in an interface. Formative think-aloud studies are used frequently in industry.

Think-aloud studies also have a range of limitations. As mentioned above, thinking aloud can influence how tasks are performed, the workload experienced during the task, and how quickly tasks are completed. This is in particular the case where participants generate Level 3 verbalization. It is also clear that thinking aloud does not produce a full and complete picture of what participants think: it requires verbalization, some thoughts might be suppressed because of the think-aloud setting, and so on. And thinking aloud has been shown to be culturally specific, which means that the information that think aloud gives might vary just due to the cultural background of the participants [153].

Paper Example 42.1.1 : Think aloud and word processing software

An early think-aloud study was born out of an interest in how people learn to use text editing systems [487]. The authors were interested in both specific issues in learning to use two different word processors and general issues and mechanisms around learning to use software.

Ten participants were asked to spend four half-days learning to use either system. The researchers where with the participants during that time to "prompt them to continue verbalizing, but did so nondirectively to avoid suggesting what they should think about" (p. 255). Participants received a self-study manual to aid in their learning, which was about how to do things with the text editing system.

Below is an example of participants' verbal protocols. One notable thing is that the experimenter (denoted with an E) asks very uniform and undirected questions to the participant (denoted P). Further, the participant gives verbal reports that are rich and hints at a couple of important mistakes about the editing system.

E: What are you thinking?

P: I'm trying to figure out what [this] exercise is supposed to be [for]—for line advance? [It is] supposed to be an exercise on errors. But am I supposed to be trying to make errors and then move back?

P: (Participant types two lines. At the end of the second line, she types a comma instead of a period and then presses return which positions the cursor at the beginning of a new blank line. Participant notices the typing error.)

P: Oh. I see. So now ....

E: What are you thinking?

P: I made a mistake up here. Now if I want to go back, I guess I would ... (looks in manual for information).

E: What are you looking at? Page 3-4? (Participant says nothing.) What is that telling you?

P: Well, I'm trying to figure out how to go back to correct that mistake. Am I supposed to correct my mistakes yet? Or am I supposed to just not worry about the mistakes? Or ... I'm going to try to go back.

P: (Participant presses backspace and incurs an error which is signaled by a beep. This is because backspace will not move the cursor beyond the left edge of the screen.)

P: Woo! It didn't like that!

P: (Participant presses correct key to move cursor up to line with typing mistake.)

E: Okay. What did you hit?

P: I pressed (identifies key).

In the analysis of these data, the authors did not code for frequencies of particular problems, but aimed to be both "clinical and inductive" [p. 258]. They mean that they try to infer the causes of the observed problems and learning difficulties. Their approach is inductive in that they try to generalize across examples. They illustrate a range of situations in which participants have difficulty learning, and hence experience frustration and have a hard time applying what they learn. They also show that participants have a hard time using the help system, in part because it is not clear to them what to ask for when they encounter a challenge. These observations were instrumental in helping researchers design better help systems, training materials, and user interfaces.

42.2. Instructions and Tasks

The instructions for think-aloud studies, including the tasks, strongly influence the results that can be obtained. Let us outline the best practices for each.

42.2.1. Instructions

The original instructions from Ericsson and Simon were to simply prompt users to "keep talking", typically after a fixed duration of silence (e.g., 30 seconds). Participants should also "think aloud as if you were alone in this room". Those instructions have been used extensively in HCI, where they are often referred to as *classic thinking aloud*. These instructions are believed to influence the participants minimally.

Alternative instructions have been studied. Boren and Ramey [87] propose a variant informed by speech genres, where the experimenter should follow the classic variant, but also (a) explain that the participant is not the object of the test, the system is; (b) ensure that the participant is the expert and primary speaker; (c) acknowledge thinking aloud by "mm hmm", "yeah", "ok".

In *relaxed thinking aloud*, more emphasis is placed on explanations and reflections. Although this is not valid according to Ericsson and Simon's account—because it is based on level-3 verbalizations—the additional information might be useful in HCI. Researchers might ask "what are you trying to achieve?" or "what are you thinking?". The reason is that explanations of interaction are useful, for instance, because they help diagnose usability problems, offer ideas for redesign possibilities, or help to understand the rationale behind a certain interaction.

The different types of instructions for verbalization have different consequences. Hertzum and colleagues [2009] compared those two variants of verbalization studies of the consequences of different variants. They found that relaxed thinking aloud affected visual search behavior, navigation behavior, and mental workload. Therefore, a seemingly minor decision in think-aloud instructs impact interaction significantly; this should be kept in mind when choosing think-aloud instructions. It is also important to remember that a think-aloud study is not an interview [327]. Users' actual attempts at doing tasks are the basis for the think-aloud study; thus, the focus is on concrete behavior and observations of that behavior are invaluable to understanding think-aloud.

Alhadreti and Mayhew [16] compared three think-aloud methods: concurrent, retrospective, and hybrid (both). The participants were asked to use a library test while using one of the three methods. They found that the concurrent method is superior to both the retrospective and the hybrid method. More usability problems were detected than when using the retrospective method, and was comparable to the hybrid method. However, concurrent methods are less time-consuming for the evaluator.

42.2.2. Tasks

The other main contributor to the validity and variability of the findings of a think-aloud study is the *task*.

Tasks may be selected to maximize realism in the sense of McGrath [518]. It is important that tasks be representative of the tasks that users would do. For instance, they may be based on tasks identified in user research (see Chapter 15). They may also be open-ended tasks that users themselves make concrete as part of the test. It is recommended to not use terms that precisely describe functionality, particular options in systems, or help participants express what they want to do in the terminology used in the system. This reduces realism because participants would have to figure this out on their own if they did not write down tasks.

For think aloud, several specific recommendations for tasks have been suggests. It is good practice to check tasks for think aloud studies in relation to these recommendations. They are as follows.

1. The tasks, in particular, the first few, should be easy. The idea is to help users learn to think aloud rather than struggle with the task. An easy first task also removes any nervousness about the study situation.

2. The tasks should be central to the users' real or imagined activities.

3. The tasks should be expressed in terms of users' real or imagined activities, not in terms of the system. The latter are called "hidden help" because the task helps users identify which feature in a system to use to accomplish the task.

4. It should be clearly stated when the task is completed. This is important because evaluators would like to know if users understand that they are done. If they do not, perhaps the system needs to give better feedback.

5. Tasks can be open-ended, so that they are in part specified by the user. For instance, rather than asking a user to book a 1-person flight from a given city, one could ask about their travel plans and use that (i.e., the city, the number of people in the party) as the task. This is more realistic and might reveal interesting usability problems when a user's particular travel requirements are not well supported.

42.3. Analysis

Analysis of think-aloud protocols has a number of steps, with implications for reliability. In *pre-processing*, the experimenter cleans the verbal data, transcribes them, and segments them according to participants, tasks, and so on. The transcribed data may be combined with data from other sources, such as video, eye tracking, or logged interactions.

In the *identification* phase, important or interesting aspects of the pre-processed data are identified and possibly classified. The purpose of separating this from transcription is that the interpretation may be made by different people and that inter-rater reliability may be assessed.

This is typically done in one of two ways. In bottom-up analysis, insights are grouped based on their frequency or prominence in the data. For instance, this may be done by affinity diagramming or thematic analysis, as discussed earlier (see subsection 11.5.3).

In top-down analysis, an existing framework, classification, or set of codes is used. The classification to use is, of course, determined by the purpose of the think-aloud study. To identify usability problems, for example, the User Action Framework has been used [312]. This framework separates different types of usability problems and helps diagnose the cause of the problem.

Think aloud studies are commonly used in *usability tests*. The goal is to identify usability problems. A usability problem may be one of three main types (adapted from Jacobsen et al. [375]):

Failure to reach goal For example: the user articulates a goal and cannot succeed in attaining it within three minutes; the user gives up; the user produces a result different from the task given; the system crashes or reaches state from which the user cannot recover.

Misunderstandings about the system For example: the user knows what the goal is but cannot pick the right action; the user expresses surprise; the user expresses ways to improve the system.

Negative experience For example: the user expresses a negative feeling; the user says something is a problem; the user expresses a negative sentiment toward the system.

The focus is then to identify such problems from the verbal data, to synthesize the types of problems across users, and to develop ideas on what to do about the problems. The usability problems are typically reported in usability problem lists, which can separate (a) the problem, (b) its causes, (c) its behavioral consequences for users, and (d) any design changes that can alleviate the problem.

In the *synthesis and implications* phase, the classified think-aloud data are used to draw implications. Such implications may be about how to fix problems; while seeing problems is easy, discovering how they can be solved can be difficult.

Paper Example 42.3.1 : The evaluator effect in think-aloud studies

The *evaluator effect* is the phenomenon that different evaluators find different usability problems; this is the case for both analytic evaluation techniques and think-aloud studies aimed at identifying usability problems (see Chapter 41). This is essentially related to the reliability of usability evaluation.

The evaluator effect was named by Jacobsen et al. [375]. They found that the same evaluators identified markedly different problems. The evaluators were told to look for *usability problems*, defined as occurrences in video recordings with the following characteristics:

What was striking about the results by Jakobsen et al. is that the evaluators agreed to a very limited degree. The figure below shows that usability problems (UPT is "Unique problem token") are not regularly found by all four evaluators of the system. Thus, only a fifth of the problems were found by all evaluators and about half of the problems were found by just one evaluator.

	1 evaluator	2 evaluators	3 evaluators	4 evaluators
Severe UPTs	8 (22%)	7 (19%)	7 (19%)	15 (41%)
All UPTs	43 (46%)	19 (20%)	12 (13%)	19 (20%)

These results cast doubts on the *reliability* of think-aloud studies and evaluations more generally.

42.4. Variations on Think-aloud Studies

The HCI field has seen many variations on think-aloud studies as they have been discussed so far. A few are worth discussing because they offer important new ways of dealing with the issues in think-aloud studies.

In *collaborative think aloud*, multiple users collaborate to use an interactive system. Rather than being instructed to think aloud "as if alone in the room", they simply talk to each other as part of interacting with the technology. Thereby, thinking aloud becomes more natural for participants. Sometimes researchers who apply this approach use techniques for classifying and making sense of data drawn from studies of conversation. In general, these approaches works well in creating unencumbered talking but face concerns about validity because they are about explanations and because they reintroduce the aspects of conversation that Ericsson and Simon tried to eliminate.

Another variant is *remote usability studies*, where users think aloud away from the evaluator.

42.5. Verbal protocols: A forgotten secret?

Think-aloud methods were covered in virtually all HCI curricula in the 1990s and early 2000s, but gradually fell out of fashion for an unknown reason. Think aloud is in a privileged position in the toolbox of evaluators. It remains the best method to obtain access to what users think and feel during computer use. Unlike other methods, such as questionnaires and analytic methods, it is *open-ended*: it allows users to express themselves without imposing a predefined taxonomy on them.

Fan et al. [233] conducted a survey study of usability professionals in 2020 ($N = 197$). They asked about their practices of using think aloud as a method. They made a surprising finding about the popularity of the method. A majority of professionals (86 %) reported using think alouds in their usability tests. The main motivation was either to inform design or to inform design *and* measure performance. Concurrent think aloud was more common (61%) than retrospective.

They also found that practices tend not to follow the best-practice recommendations of Ericsson and Simon [225]. A majority of 61% of respondents "almost never" asked their users to practice think aloud before entering the study. Moreover, practitioners often posed leading prompts to users, such as asking them to talk about their emotions. Improving on rigor is something where easy gains are achievable.

To this end, the authors conclude with recommendations for think aloud studies:

1. Practice sessions should be conducted to 'warm up' users and get them verbalize more often.
2. Instructions on what to report should be neutral (and not leading).
3. Evaluators should not interrupt the participants during verbalizations.
4. Think alouds are not limited to lab studies, but can be done in remote usability tests. (Remote usability tests are carried out over a virtual connection, for example using a remote desktop feature and video conferencing.)
5. Improve the efficiency of data analysis to increase the value of think alouds. This can be achieved by developing reusable coding schemes, using machine learning methods for speech recognition, and interactive tools for labeling.

Finally, it is worth nothing that recording think-aloud studies can have an impact on stakeholders. Usability problems become relatable when seeing a short video of a struggling user and talking aloud. In this regard, think-aloud studies trump most other evaluation methods.

Summary

- Think-aloud studies give an insight into users' thinking processes and may be used to infer usability problems.

- Think-aloud studies are persuasive and inexpensive, although the evaluator effect suggests that reliability may be low.
- Following best practices increases the cost-efficiency and reliability of the method.

Exercises

1. Task selection. Selecting tasks for think-aloud studies. Find a selection of tasks made for usability studies; the cooperative usability evaluation studies (CUE) show many. The data are available at http://www.dialogdesign.dk/ CUE.html. Discuss them.
2. Conducting a think-aloud study. Select a website (or application, or other type of user interface) that helps people plan trips. Plan a think-aloud study for the website, focusing on creating a set of suitable tasks and the instructions to give the participants. Then conduct the think-aloud study. Reflect on what was hard and easy.
3. Comparing approaches. Molich [538] gives an overview of comparative usability studies, a series of studies that compare the results of usability evaluations (including evaluations performed by professionals using think-aloud tests). Pick one of the studies using think aloud and compare the variation in one aspect of the planning, running, or analysis of the study. What works well and what does not work well? What do you think might impact the test?

43. Experiments

Imagine that you have developed a new version of a search feature for an operating system. Users can click a magnifier glass icon and type in a search query, results would be listed underneath. You are certain about the benefits of the new design, but you need something to convince other people. What kind of test could show, convincingly, whether users actually like the feature better than the previous version? You could ask people to use the new feature and inquire if they like it. But that would not help you relate it to the previous search feature; you would not know if it is better. You could ask users of the old feature and users of the new feature to rate how well they liked the respective search features and compare those ratings. But even if a difference was found, an inconvenient alternative explanation would be there: Perhaps the users in the two groups were different, for example, in their experience or age, and that would illustrate the difference. You could also seek expert opinion, consulting colleagues and HCI researchers to learn which search feature is best. However, this may be difficult to assess, even for experts. In the worst case, it might turn into a clash of opinions where everything is trumped by the HIPPO, or the highest-paid person's opinion (see [420]). What you need is a method that allows you to firmly attribute an observed difference to the new search feature and nothing else. That method is called an *experiment*.

An experiment is "a study in which an intervention is introduced to observe its effects" [738, p. 12]; see also Figure 43.1. An experimenter changes something, or intervenes, while keeping everything else the same, and observes the effect of the change. An experiment is a deliberate change in circumstances: The experimenter imposes some condition or constraint in it. Such intervention may be of a variety of kinds; in HCI it is often a technology, but could be different kinds of training, user groups, use situations, or tasks. In common practice, an intervention is designated as a level of treatment (e.g., comparison of user interface designs), group (e.g., comparison of two age groups), or condition (e.g., comparison of different instructions to users).

The design of experiments boils down to defining an intervention and what is being measured. *An experimental design* associates variables defining the intervention (independent variables) and what is being measured (dependent variables). Something that is systematically varied in an intervention is called an *independent variable*. Consider, for example, changing the color of a button as one independent variable, or the age group of the user. The effects of the intervention are measured as *dependent variables*, those that depend on the intervention. For example, one could measure task completion time or errors. In HCI, measures are often related to usability or experience of the technology.

If the relationship between the dependent and the independent variables was fully under the experimenter's control, the observed changes in the dependent variables could be attributed to the intervention, and to that only. However, experiments with human

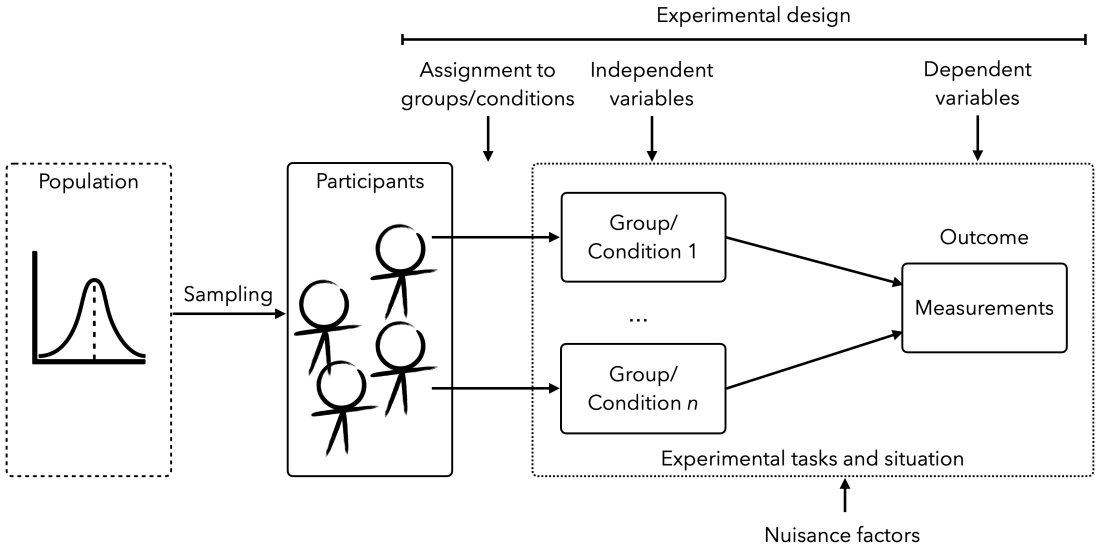


Figure 43.1.: The main components of experiments.

participants need to deal with a plethora of *other* factors besides the independent variable. Image a study where users were first asked to carry out a task A with a user interface and then task B. Here, the order of the interventions interacts with users' learning. Any measurements in task B would be affected by what users learned in task A. In general, such factors influence the situation under study, and thus potentially affect the dependent variables. Such factors are called *nuisance factors*: a nuisance in the sense that a factor like this threatens the attribution of cause to the intervention. Was it the task or was it learning that caused the observed difference? Experimental designs have many ways to deal with nuisance factors: controlling such factors, holding them constant, or distributing them randomly across levels of the independent variable. Consider, for example, getting rid of the effect of users in a study having seen the old features in their work. If we would like to get rid of that effect, we talk about controlling it.

Finally, the choice of interventions and measurements must not be arbitrary. *Hypotheses* are statements that connect variation in independent variables with expectations about variation in the dependent variables. Would you expect your new feature to be better than the baseline design in usability; and if so, why? Explicating hypotheses is critical for high quality evaluations. Hypotheses can avoid being fooled by observations subject to noise and error, to avoid being biased by one's own intuitions, and to avoid second-guessing.

Note that the above definition excludes the understanding implied in some common usages of the word experiment, including that of "trying something new" or "an innovative act or procedure". In contrast, experiments in the context of evaluation aim to *establish causal conclusions about which factors influence a situation.* Experiments try to rule out alternative explanations besides the factors being manipulated. This chapter concerns how to enable such conclusions to be drawn, in particular about the use of interactive computing systems. The following subsections explain these components of the experiments in more

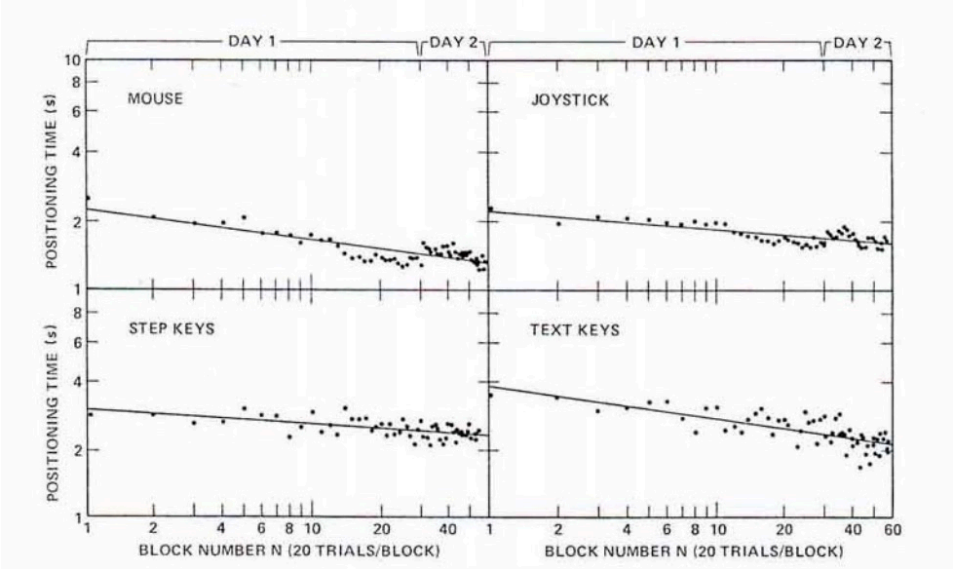
detail; the box below summarizes an early and influential experiment.

Paper Example 43.0.1 : Development of an experimental paradigm for evaluating input devices

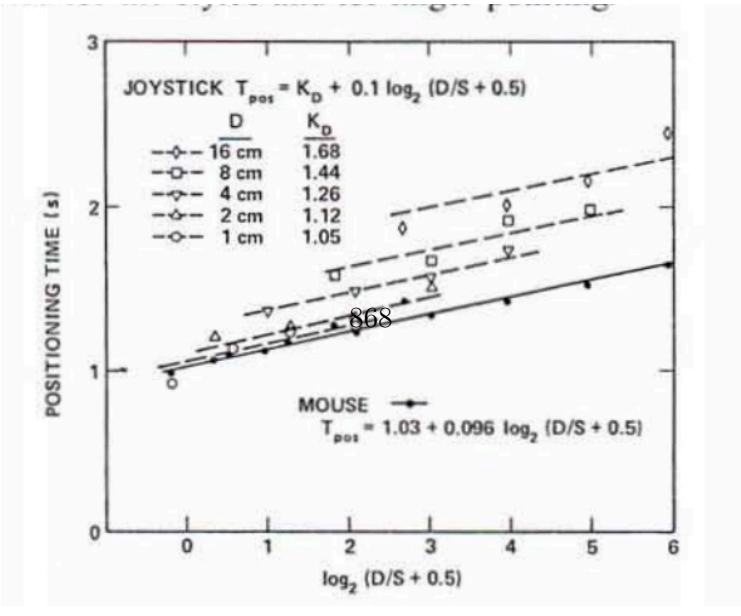
In 1978, Card et al. [128] reported a now classic experiment with input devices. The study contrasted the efficiency of input techniques for selecting text. At that time, such comparisons were rare; the paper was among the first in HCI reporting experiments involving a mouse and using Fitts's law [493] (see Chapter 4).

The authors had five participants use a mouse, a rate-controlled isometric joystick, and two variants of keys; participants used all four. In the experimental task, the participants had to select highlighted text at varying distances (1 to 16 cm) and sizes (1 to 10 characters). Participants used each device until they did not improve in performance. They did from 1200 to 1800 selections for each device, or four to six hours of pointing.

This experimental design allowed Card and colleagues to plot the development in learning to use the device. They did so in the plot below, which illustrates a power low of practice: $T_n = T_1 - n^{-a}$, where T_n is the time used at trial n and a is a constant. In a log plot this shows as a line whose slope indicates the learning rate.



The authors also use the experiment to plot the positioning time (their main dependent variable) against the index of difficulty – a measure combining the size of targets and the distance between them, two independent variables.



43.1. Research Questions

Planning an experiment, even for evaluative purposes, should always depart from the *research questions* one wishes to address. A research question is a "stated lack of understanding about some phenomenon in human use of computing, or the stated inability to construct interactive technology to address that phenomenon for desired ends" [608]. In short, experiments are motivated by *knowledge gaps*. For example, in the example starting this chapter, this knowledge gap concerned the effect of the new search feature on usability.

More broadly, research questions can be divided into three classes: (1) empirical questions: phenomena and effects in human-computer interaction, (2) constructive question: the ability to construct systems and designs with desirable properties, and (3) conceptual questions: relationships between theoretical constructs that represent interaction. Experimental research can serve all three. In addition to empirical questions, we can conduct an experiment with the purpose of setting objectives for design (constructive), or to distinguish between competing theories (conceptual).

But how to come up with a *good* research question? One can think about this by starting form the opposite: a poor research question. Let us entertain two common *objections* to a finished and written up experiment: (1) "so what" and (2) "no surprises" [236]. The "so what" objection suggests that the imagined results of an experiment should be interesting and nontrivial; they should matter to theory or practice. Even if running and analyzing the experiment proceeds as imagined, will people find it interesting? Will it add to our understanding of HCI in important ways? Every so often, this objection is voiced by reviewers as "this is not significant", meaning that while the findings are novel and valid, they do not add to the research literature in a substantial and important manner.

The "no surprises" objection suggests that the results should add to or depart from what we already know; they should not be predictable given earlier studies. One should not do an experiment if the results are clear in advance. For instance, if a simple predictive model shows a user interface superior to another or if a technology is without doubt superior to an alternative, then the experiment does not have the possibility to surprise us. Sometimes, of course, new technologies, use situations, or user groups may make it hard to know if earlier findings or theories apply. The "no surprises" objection may be raised both because the experimental setup is biased (we will discuss how to avoid this later) and because the results are easily predictable from the literature (we will also discuss how to avoid this later). Most importantly, both of these objections should be considered *before* deciding to run an experiment.

The experimental method is only one of the many approaches available in evaluation. The decision to choose should be based on careful consideration of the pros and cons. And this boils down to the research questions. Following the discussion of McGrath [518] in Chapter 10, one reason is that experiments maximize precision, but this is achieved at the expense of generalizability and realism. Experiments also allow precise manipulation of tasks and settings, as well as detailed data collection. This allows us to understand *specific mechanisms* involved in interaction. Experiments also allow us to control external

factors that may be hard to exclude by other means. For example, if you want to evaluate user performance of a mobile application, you may want to prevent notifications and multitasking during your measurements, because they would add noise to the data. They also allow us to collect fine-grained data about how users behave with an interface.

Laboratory conditions permit the use of measurement devices such as eye-trackers, motion trackers, and physiological sensing like electromyography (EMG), which annot be easily deployed outside laboratory conditions. If these qualities are important for your research questions, the experimental method is well suited. Another reason is that experiments allow us to investigate research questions about the use of technology without actually deploying it. This is valuable when a prototype does not have all features a real product would need. Experiments are also efficient: they allow us to 'compress time' and study phenomena that occur infrequently. For example, you can arrange a laboratory study that goes through 5-10 tasks within an hour, while the chances of those occurring in real-world use might take days or weeks. Finally, in experiments, we can organize circumstances where we can safely study events that would otherwise be unethical because of causing harm to participants. Research questions along these lines are suitable for experiments.

Another important reason for experiments is the egocentric fallacy [443]. According to this fallacy, we tend to overestimate the power of our own intuition of human behavior. However, intuition is weak in discovering latent (unobservable) mechanisms behind our behavior and experience. The mechanisms behind human behavior are beyond intuition. At the same time, we underestimate the extent to which we differ from other people. This is particularly problematic in HCI, where a number of technologies that we propose will have been developed and iteratively refined by ourselves or by close collaborators. Experiments help overcome this fallacy.

In the same spirit, there are also some research questions for which experiments are ill-suited. They rarely work well for studying how people decide to act with technology in real circumstances. See instead the chapters on interviews (**??**) or field studies (Chapter 12).

43.2. Research Hypotheses

Research questions may be further elaborated as *research hypotheses*. Hypotheses are statements that link manipulations of the independent variables to differences in the dependent variables. For example, Nass et al. [565] hypothesized that "subjects will perceive a computer with dominant characteristics as being dominant" (p. 288). Gutwin and Greenberg [301] held the hypothesis that "better support for workspace awareness can improve the usability of these shared computational workspaces" (p. 511). Hypotheses are important also in evaluative studies. However, creating good hypotheses is hard.

Good hypotheses are (a) testable, (b) concise, and (c) name key constructs. The first example given above is testable because one may compare computers with and without dominant characteristics and expect a significant difference in participants' perception of dominance. That example names the key construct dominant, both as something that may be manipulated in computer interfaces (an independent variable) and as something that

participants perceive (a dependent variable, assessed, for instance by a questionnaire).

There are many benefits of formulating research questions as hypotheses. First, hypotheses help gain clarity about what one is doing and may help focus a research question. Second, formulating hypotheses helps one think through what earlier work says about the experiment being designed. Third, hypotheses help report an experiment. Fourth, hypotheses are tied to theory. They help think through explanations in advance and allow for genuine surprise about the experimental results.

Not all experiments, however, need hypotheses. The questions that drive the experiments fall in two broad groups, sometimes summarized as "testing theory" and "hunting phenomena" [258]. In the former group, there are clear expectations about the outcome of the experiment, typically build on predictions from earlier work. This is often formulated as hypotheses, statements that link the levels of what the experiment is manipulating to outcomes in the measured variables. In the latter group, the experimenter holds less clear expectations and holds more open-minded curiosity. These are also called *explorative* experiments.

43.3. Independent Variables

Independent variables refer to the types of events or factors that we want to draw causal conclusions about. Independent variables can be about anything we can systematically control. They can be about the types of users (e.g., novice, intermittent, expert), types of user interface (e.g., command-line, graphical), form of instruction (offline, online), the type of feedback, and so on.

In selecting independent variables, it is important to remember that the experiments are carried out to *gain information*. Results should not be obvious in advance, and experiments should not be set up to generate winning conditions and losing conditions.

To ensure that we do not pick arbitrary IVs, the choice of independent variables should follow from stated research question. However, that there are many traps in bridging the two. For instance, let us consider again the example from the first part of this chapter (page 865). Recall that we wanted to compare two versions of a search function. In doing so, we need to establish what is considered belonging to the search function: do we want to include the highlighting of search results in our study? Do we want to consider search results opening to a context menu underneath the search button or in a new window, or both? What if the new search function has a case-sensitive option whereas the old does not? For each of those questions, a careless experimenter may invalidate the experiment.

43.3.1. Levels of an independent variable

The *levels of the independent variable* are all possible values that an IV can take in an expeeriment. For example, if your study compares three UI features, the independent variable 'UI feature' would have three levels. If you compare two systems, 'the system' would be one IV with two levels (system A and system B).

43.3.2. Eliminating confounds

One difficulty lies in making levels of the variable (say, two versions of an interface) similar in all essential aspects, except that one is manipulating. If the interfaces one is comparing are not similar, the effects one is studying may be confounded by the dissimilar features. The concerns relating to making conditions similar require an experimenter to ensure that:

- all non-essential aspects are similar. So for instance, if search is supported, it should be across all conditions. If shortcuts are available, it should be in all conditions;
- the screen real estate is similar across interface;
- the training and the users' skill with the variants of interface is similar;
- the setting in which the interfaces is used is similar;
- comparable information available in the interfaces;
- comparable hardware is used (e.g., for input and output);
- the time allotted and the criteria for success are similar across levels of the independent variable.

43.3.3. Selecting meaningful baselines

Another concern is to ensure meaningful baselines. A "strong baseline" refers to a solution that is considered best on the market or literature, the state-of-the-art alternative. This could, for instance, be an interface that implements the typical way of performing a task. Our concern here is to ensure that the baseline (or control) is as strong as possible. Munzner [551] discussed what she termed "Straw Man Comparisons", that is, cases where authors compared their interfaces against outdated work, rather than the state-of-the-art approaches. Although Munzer wrote about information visualization, studies in HCI more generally are sometimes done by comparing novel interfaces with weak or incomplete versions of the current state of the art.

43.4. Participants

The participants in the experiments are the people whose interaction with technology we want to study. They should be seen as a representative sample from the group we want to draw conclusions about. The selection of participants impacts which conclusions can be drawn. It also shapes the practicalities of running the experiment.

The key question is *who* should participate. Representativeness is important. Recruiting a convenience sample – whoever happens to be available – should be done with caution. Barkhuus and Rode [45] found that about half of a sample of studies from the CHI conference used students. Increasingly, participants are also recruited online, for instance, through Amazon Mechanical Turk. The participants hape what happens in the

experiment as well as how well we may generalize findings. We may choose experienced or inexperienced computer users; we may find domain experts or novices. Thus, they should be selected so that we can validly answer the research question we are interested in.

Another important question is *how many* should participate. Typically, the number of participants in studies in HCI is around 12 [119]; for controlled experiments where participants are present in person, the number is about 20. However, crucially, this does not mean that 12 is enough for *your study*, only that this number is about the average across most of the published literature.

The principled way of finding an appropriate number of participants is to do *power analysis* [162]. Power analysis helps estimate the probability that one detects a difference in dependent variables between the levels of the independent variable if one knows (or can qualify a guess about) the magnitude of the effect one is examining. Power analyses are often a depressing reading. Typically, many participants are required to achieve a reasonable power; say, an 80% probability of finding a difference. To detect medium-sized differences between the two conditions at this probability, one would need 64 participants in each condition in a between-subjects experiment. Medium-sized effects found in the HCI literature include differences between broad and deep menus, or between selection with mouse and keyboards. There are tools to help with such analysis (e.g., G*power).

43.5. Research Ethics

A key consideration in experimental research is the ethical treatment of the participants. They should in no circumstances be harmed. Participating in the study should not have negative consequences on their lives. The principles for the ethical treatment of participants in behavioral research have been established, the Helsinki Declaration on Ethical Principles for Medical Research Involving Human Subjects; recent updates include the American Psychological Association's Ethical Principles of Psychologists and Code of Conduct (http://www.apa.org/ethics/) and the code of conduct for the ACM (https://www.acm.org/code-of-ethics). Research organizations have formal guidelines and requirements in place to ensure respectful, legal, and ethically defensible experiments.

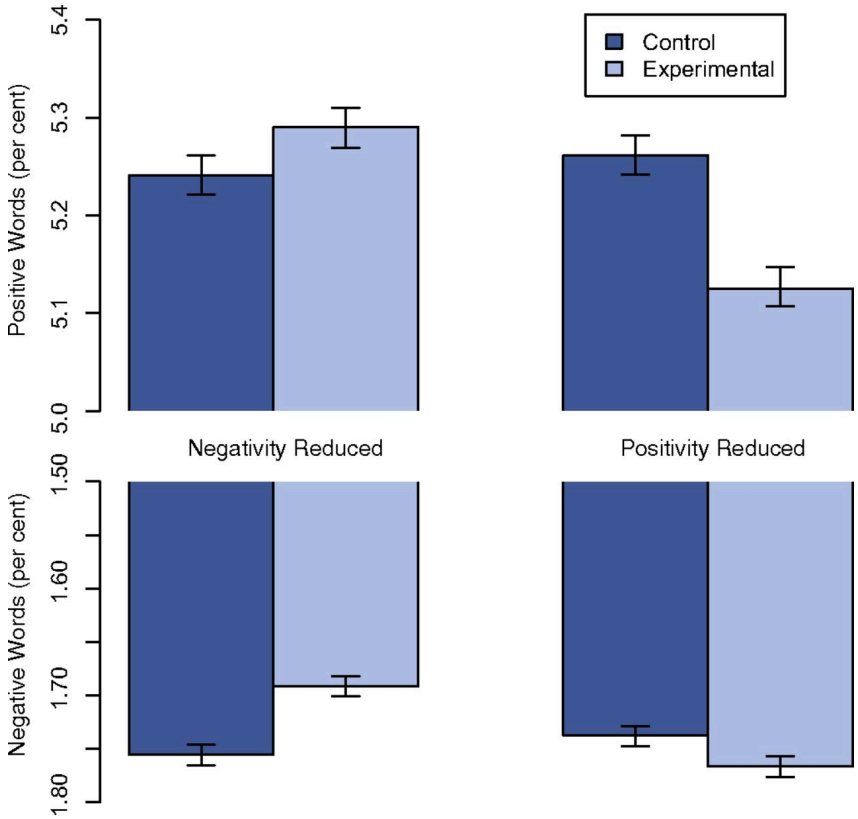
Key goals for ethical experimentation include:

- Treat participants with respect: value their time, honor their opinions, take any criticism seriously.
- Do not expose participants to dangerous or potentially harmful situations; this includes physical, mental, and emotional concerns.
- Make sure participants want to participate; get informed consent (there are templates available online and your institution might also offer one).
- Make sure any reimbursements to participants are adequate: they should not make it necessary for participants to enroll in your study but still compensate for their time and any expenses incurred by participating (e.g., transportation costs).

- Make sure to debrief the participants, explaining them about the purpose of the experiment and taking any questions they might have about your research.

Paper Example 43.5.1 : Controversial Facebook experiments

In 2014, it was reported that a large-scale experiment had been conducted on Facebook which attempted to manipulate the emotions of its users [424]. Close to 700,000 Facebook users had their feed of stories manipulated so that they experienced a reduced amount of emotional content. The figure below shows how the manipulation (negativity reduces among one's friends or positivity reduced) change the positive and negative words that users post subsequently (the dependent variable). This figure shows a clear impact of the experimental conditions compared to the control conditions (no change in emotional content. This results are significant because it shows that emotional contagion, having your emotional state changed based on the people that surround you, can happen on social networks and without any direct interaction between people.



However, the paper was controversial. The Facebook users who were part of the study did not give informed consent as would be normally expected in research and did not, in a clear manner, give permission to have their emotions manipulated experimentally.

43.6. Experimental Design

Let us consider again the search functionality example we started with. If you conducted a study where the only functionality being assessed was the new feature, you could not draw a conclusion on whether it improved over the original or not. You would not have a baseline to compare against. This is an example of why we need to introduce *experimental conditions* that allow us to answer our research questions. The variables to which we map those conditions are called independent variables. They are called independent because the observations collected in those conditions are independent of each other. This is ensured by allocating participants either systematically or randomly to those conditions.

Experiments in HCI almost always involve more than one level of an independent variable, for instance two alternative interfaces for a task, a range of different instruction materials, or different approaches to delivering notifications. The assignment of participants to the levels of the independent variable is the main consideration in experimental design. The aim is to ensure internal validity, that is, the ability of an experiment to attribute differences observed in the dependent variables to manipulations of independent variables [738]. Furthermore, experimental design need to consider subsequent running and analysis of experiments. Simple designs are typically easier for participants; the statistical analysis and interpretation are easier for experimenters.

One key decision is whether participants experience all or just one level of the independent variable. The former type of design is called *within*-participants, because the independent variable is varied for each participant, the latter type is called *between*-participants.

In within-participants experiments, participants serve as their own control. So even if a person varies in some trait or behavior, a so-called wild-card participant, that variation is cancelled because the participant uses all levels of the independent variable. However, those types of designs are not without problems. For instance, within-participant experiments suffered from learning effects. Participants may learn about the interface or the task, and therefore their experience or performance might change during the experiment.

Between-participant experiments have many benefits. They are easy to analyze and they do not suffer from the possibility of influence across conditions because participants use just one level of the independent variable. The key drawback of between-participant experiments is that they cannot control for individual differences and therefore require more participants.

As suggested in Figure 43.1, other factors in experiments may influence the experimental situation. There are several ways to deal with these. The workhorse of experimental design is *randomization*; this is often cited as a defining characteristic of experiments. Randomization means that participants are assigned to conditions at random. Thereby, the influences of other factors than those being manipulated are randomly distributed over conditions. One of the authors of this book has the motto "when in doubt, randomize". *Control* means simply restricting the variable to one level; one may experiment, for instance, with only left-handed persons. Thereby, the influence of handedness can be ignored in the analysis of the experiment.

Sometimes experiments have more than one independent variable, which complicate

their design. For instance, an experimenter may want to study three interfaces (say, a command line versus a graphical user interface versus a non-computer method). There are easy ways to combine such variation in independent variables, called Latin squares. If we use the first letters of the interfaces (C, G, N) as shorthand, we could have one use do C, then G, then N. Another could do G, N, C. Another, N, C, G. That organization protects against a number of issues in the experimental design by having an equal number of users use each interface first, second, and last (you can see this from Latin squares in that the count of interfaces in each column is the same). More complex ways of setting up experiments. For instance, if we want to compare the interfaces just discussed across three ways of training (hints vs paper manual vs integrated manual), then we can use a Greco-Latin square. It could lead to us organizing the experiment as follows:

```
          1    2    3
User A: C+H  G+P  N+I
User B: G+I  N+H  C+P
User C: N+P  C+I  G+H
```

Again, the sum of types of training is similar across columns and each combination of training and interface occurs just once. Generators for Latin and Greco-Latin square designs may be found online.

43.7. Dependent Variables

In experiments, the aim is to understand how the independent variable influences the interaction. By convention, we call measures of this influence *dependent variables* because they depend or result from our manipulations of the independent variable. Another way to think of dependent variables in HCI is that they indicate the quality of the interaction numerically, say with the duration or accuracy of the interaction.

A crucial question for dependent variables is *conceptualization* [738]. Conceptualization concerns making the meaning of concepts in an experiment's research questions clear, defining them precisely, and separating different dimensions of meaning. For instance, while the learnability of a user interface is (superficially) easy as a dependent variable, defining it is much harder. Grossman et al. [291] showed how the literature displays many understandings of learnability. If an experiment does not clearly conceptualize learnability, the validity of any inference from that experiment may be reduced because learnability may mean many different things. Similarly, task completion time is easy to measure in many experiments. But it may not be the best conceptualization of quality of an interface. For instance, studies vary in whether they see low task completion times as good (minimizing resource expenditure) or bad (expressing a lack of engagement); see Hornbæk et al. [351]. Unthinkingly measuring task completion time therefore reflects too poor a conceptualization of quality. Another example is the notion of user friendliness, which has been used in HCI for decades. On the surface, it may seem like a natural ingredient of a research question and therefore as a dependent variable. But it is difficult to define and it is hard to separate its dimensions. Conceptualization shows that it is a dead end for many experiments.

Several tools of thought help conceptualizing dependent variables. First, one may refer to the models and findings in previous chapters on user experience, usability, performance, and collaboration. Each of these represents important concepts that we may use to make research questions more precise. For instance, the discussion of independent dimensions of usability suggests that we should consider whether we are thinking of effectiveness, efficiency, or satisfaction when we are experimentally trying to find out which of a a set of interfaces that is more usable. Or perhaps we need to consider all of them. Second, Newman and Taylor proposed to think about the critical parameters of a certain situation [571]. A critical parameter is a performance indicator that captures aspects of performance that are critical to success, which is domain or application specific, and which is stable over variations of interface. Part of the challenge in applying catalogs of measures is to ensure that at least some measures chosen are critical in the above sense (and not just generic time or error measures).

A second crucial question for dependent variables is *operationalization* [738]. Operationalization is about turning the concepts that our research question name into something we may measure. The main consideration concerns the extent to which the actual measures collected reflect what the experimenter wishes to measure, or whether it is possible to make "inferences from sampling particulars of a study to the higher-order constructs they represent" [738, p. 65].

We may ponder the following questions in operationalizing dependent variables. First, how will we actually obtain the measure. Second, use validated measures and questionnnaires. Third, multiple measures of the same construct increase reliability and strengthen the validity of claims about constructs. Using just one operationalization of the construct faces a mono-method threat to validity [738]. It means that we are more prone to not measuring what we think we are measuring if we use just one indicator for a construct.

Most measures in research studies in HCI are task completion time, accuracy or error rates, and questionnaire answers. However, a couple of additional types of data are worth collecting in experiments. One important type is about the *interaction process*, for instance, which commands participants activate or how they move their mouse. Such data may help us understand the interaction process (rather than just the outcome) and may help us think about why something happens in an experiment.

Whereas dependent variables need to be numeric, many exemplary experiments also collect *qualitative data*, for instance in the form of interviews and observations. Some experiments also rely solely on qualitative data. For instance, O'Hara and Sellen [598] reported a much-cited experiment on reading from paper and from a computer. While they used an experimental setup—using for instance random assignment of participants to either paper or a computer condition—they only reported qualitative data on reading strategies and activities that differed between paper and computer. Such data is valuable when experiments go well (as in O'Hara and Sellen's study), but it is also useful in understanding why an experiment failed.

43.8. Experimental Situation

Experiments put participants into *experimental situations*: the particular circumstances where they are asked to carry out tasks. The design of these situations matters.

One decision concerns the *activities* that participants will engage in. They are often prescribed as tasks. One may select tasks in many ways. One is to select tasks that are representative of what users would do outside the experiment. Munzner [551] discussed selection of tasks in information visualization and wrote 'A study is not very interesting if it shows a nice result for a task that nobody will ever actually do, or a task much less common or important than some other task. You need to convince the reader that your tasks are a reasonable abstraction of the real-world tasks done by your target users " (p. 147). One way of ensuring representativeness is to use tasks that users have been observed doing.

Another approach to selecting tasks is to use simple tasks that capture the essence of what is being investigated. The idea is to reduce variation and remove non-essential features of a task; this idea is similar to the approach for selecting independent variables that was earlier referred to as essential features. For instance, many studies of pointing techniques use the ISO multidirectional tapping task. This task type requires participants to tap circular or square targets arranged in a circle. It does not represent pointing in the wild, but is widely accepted as a useful task for experiments.

Another decision about the experimental situation concerns whether the experiment take place in the lab or in the field. In lab experiments, the setting is controlled and the effect of external influences minimized. In field experiments, the setting is real, although the experimental manipulations are instigated by the experimenter. The view taken here is that neither choice of setting is better than the other; rather, they have relative benefits and drawbacks. We discuss field experiments in Chapter 40.

43.9. Analysis and Interpretation

What should one do with the collected data? The naive approach would be to take means of data in different conditions and compare them. What could go wrong with this? If users report an experience of 3.5 in condition A and 4.2 in condition B, is this not a sufficient basis to conclude that the one is better than the other?

Perhaps the most critical piece of knowledge in analysing data concerns *variance*. Every observation in an HCI experiment is susceptible to variation. Observations are compromised by variation in repeated attempts of users, our measurement instruments, and various random effects. If variance in the two conditions A and B is large enough, it could be that the means differ because of chance. In other words, if you were to repeat the experiment, the result could be different, even flip.

Statistical analysis gives us rigorous tools to understand what we can conclude from data. Statistics is the science of drawing valid conclusions from datasets. In HCI, we use statistical analysis for different purposes, including:

- Exploring and learning about the distribution of variables or their relationships;

- Describing relationships between independent and dependent variables;
- Testing if relationships describe reliable differences in the population from which the sample was taken;
- Identifying a factor that caused or contributed to an observed effect;
- Testing if a statistical model accurately describes the dataset.

In this book, we do not offer a comprehensive overview of statistical methods for HCI. We here review some of the more popular frequentist methods and thinking behind them. For a more thorough treatment, see the book on the topic by Robertson and Kaptein [682]. Recent research has also looked at Bayesian methods for statistical testing, such as the Bayes factor. An in-depth treatment of frequentist and Bayesian statistics can be found in Cox [171]. Note that we discuss the analysis of qualitative elsewhere in the book: In chapters on interviews, think-aloud protocols, and conversational analysis.

Statistical analysis is divided into two main classes according to purpose:

- Descriptive statistics, where the goal is to describe relationships between variables in the dataset;
- Inferential statistics, where the goal is to draw a conclusion about the population from which the sample was drawn.

It is a good practice to start analysis by describing the dataset. *Descriptive statistics* refers to the use of summary statistics, like graphs, tables, and models, to describe a set of data. For example, imagine you had collected data on accuracy and speed of pointing with two input methods A and B. What you should do is to plot the distributions of the two dependent variables for the two methods. What could you learn? For example, are the distributions normally distributed, are they skewed, how much variance is there? Based on this information, you could produce *summary statistics*, such as mean, median, min and max of each variable. *Graphs* can then be used to visualize them further, for example histograms, scatter plots, line plots etc. Bivariate graphs show relationships between two variables, trivariate among three, and multi-variate more than that.

Inferential statistics refers to the attempt to generalize observations in a sample. A distinction is made between the set of observations and the population it comes from. For example, in the case of the search functionality example, one may be interested in estimating what task completion time is for regular users (population), not just the one recruited to the study (sample). Obviously, if the sample is not *representative*, conclusions drawn based on it can be flawed. There are many reasons why the sample may be unrepresentative. For example, we often use university students to represent regular users. However, they may differ in many respects: age, socio-economic and background, etcetera. This is why it is important that participants and tasks are sampled such that they represent the population. There are several strategies to ensure that: random sampling, stratified sampling, and systematic random sampling, for example.

Inferential statistics may also start by plotting. *Confidence intervals* provide estimates for the range of values that we think the true population value should fall in. For example,

if the 95 % confidence interval of our task completion time variable was [14.5, 17.9], it would mean that we are 95 % confident that the true population value was between 14.5 s and 17.9.

Statistical testing refers to testing if a difference exists between conditions. Since we are talking about inferential statistics, we are not interested whether the difference exists in the dataset, because there almost always is *some* difference, but whether this represents a true difference in the population. Here, the research hypotheses need to be translated into statistical hypotheses, which can then be tested.

Multiple methods exists for statistical testing. They can be divided into two main groups: parametric and non-parametric. Parametric tests use parameters to describe the population. They make distributional assumptions about the population, which must be first checked in order to proceed to use the corresponding test. The most commonly used parametric tests are t-test and its generalization ANOVA. *Non-parametric tests* make no assumption about the underlying distribution, which makes them more flexible as a class of tests.

Regression models help us understand the nature of relationship among two or more variables. In a regression model, we try to understand the relationship between *predictor* and *response* variables. Typically predictor variables would be our independent variables, and response variables the dependent variables. However, any *covariant*, or uncontrolled but recorded variable, could be included, too. For example, we could use regression to find a relationship between age and task completion time. If the p-value of the regression was below an apriori threshold (often 0.05), we can conclude that there would be a significant trend.

43.10. Hypothesis Testing

A very common approach to analyze experiments in HCI is through *hypothesis testing*.

43.10.1. Statistical significance testing

The high-level logic of a hypothesis testing is as follows. Assume you have a sample X with n measurements, $X = \{x_1, x_2, \ldots, x_n\}$.

We now assume that this sample was drawn from a particular distribution, let us call it N. We call this assumption the *null hypothesis* and it is typically denoted H_0.

We can now ask whether our sample was indeed drawn from a particular distribution N and we do this by asking whether it is possible to *reject the null hypothesis*. This means that we are asking whether we can, with some probability, state that the sample *was not drawn* from the distribution N.

While not a typical significance test in HCI, let us consider the distribution N we are interested in to be a standard Normal distribution $N(x) \sim N(0, 1)$. We can then ask whether an individual measurement x in our sample is drawn from N. In this case, this means x should not deviate much from 0. If a measurement x in our sample is large, say 3, then the probability that it is drawn from a standard Normal distribution is very low. However, if a measure x in our sample is small, say 0.3, then the probability is quite high.

This can be readily realized by considering that the range $|x| \leq 1$ occupies 68% of the probability mass of a standard Normal distribution. In other words, the higher x is, the more confident we can be that x was *not* drawn from a standard Normal distribution.

In hypothesis testing we define a criterion value for determining whether the probability that x is not drawn from N justifies rejecting the null hypothesis. Common criterion values are 0.05, 0.01, and 0.001. These criterion values are called *confidence levels*. For example, if we set the confidence level to 0.046, then we would reject the null hypothesis for a measurement x if x different from the value $x = 0$ by 2. That is, for any value of $x = 2$ or higher would give us cause to reject H_0 at significance level 0.046.

Statistical significance tests perform this type of hypothesis testing. However, they also take into account additional factors, such as several measurements in the sample, and they consider more appropriate distributions than a standard Normal distribution.

43.10.2. Example: between-subjects analysis of variance

Assume there is a difference in the measurements we obtained via the dependent variables when we manipulated the independent variables. This difference can be due to two things.

1. Our manipulation of the independent variable.
2. *Error*, which in this context means there is no true difference. Instead, the difference we measured was just due to random chance.

Significance tests help us decide whether measured differences are statistically significant, meaning that we can be reasonably confident that the difference is not due to chance but due to our manipulation of the independent variables.

Now assume we have sampled two groups from a user population and we exposed each group to a different method, say Method A and Method B. The *method* is then our independent variable with two levels: Method A or Method B.

We now believe a right way to compare these methods is to investigate if the means of the measures we have taken differ between the two groups. The null hypothesis H_0 says that for some predetermined confidence level there is no actual difference between the means and any measured difference is solely due to sampling error. If we reject the null hypothesis H_0 then we have a significant result at said confidence level.

We will demonstrate statistical significance testing in HCI by using a method called one-way analysis of variance (ANOVA). It is a significance test used to determine whether two sample means are significantly different in the statistical sense. The sample means must have been generated from a between-subjects experimental design.

The statistical term *error* is the amount an observation differs from the population mean. Typically the population mean is *unobservable*.

The statistical term *residual* is the amount an observation differs from the sample mean. Unlike the population mean, the sample mean is *observable*.

Assume we have obtained samples from a Normal distribution: $X_1, X_2, \ldots, X_n \sim N(\mu, \sigma^2)$. Then the sample mean is:

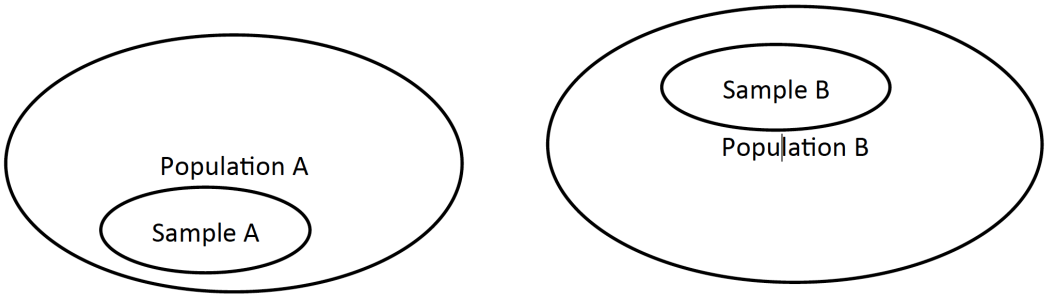


Figure 43.2.: An illustration of the acquisition of two samples from two populations.

$$\bar{X} = \frac{X_1, X_2, \ldots, X_n}{n} \tag{43.1}$$

and the error is $e_i = X_i - \mu$ and the residual is $r_i = X_i - \bar{X}$.

Now, *why* would there be a difference between the sample means of group A and B (Figure 43.2)? There are two possible reasons:

1. *Because of* group membership. This means the difference is due to an effect of the independent variable on the dependent variable.

2. *Not because of* group membership. This means the difference is merely due to sampling error.

The logic of ANOVA is as follows. There are two independent estimates of the population variance that can be obtained: (1) a between-groups estimate, which is the effect of the independent variable *and* error; and (2) a within-groups estimate, which is just error.

Our null hypothesis H_0 is that the two populations A and B have equal means:

$$H_0 : \mu_A = \mu_B \tag{43.2}$$

Given H_0, the between-groups and within-groups variance estimates should be equal. This is because H_0 assumes the effect of the independent variable does not exist. Then both variance estimates reflect error and their ratio is 1. A ratio larger than 1 suggests an effect of the independent variable.

A *sum of squares* (SS) is simply the sum of the squared residuals:

$$SS = \sum_i \left(X_i - \bar{X}\right)^2 \tag{43.3}$$

Now let us consider the sources of variability in a between-subjects analysis of variance. The total variability *within* Sample A is:

$$SS_A = \sum_i \left(X_{Ai} - \bar{X}_A\right)^2 \tag{43.4}$$

The total variability *within* Sample B is:

$$SS_B = \sum_i \left(X_{Bi} - \bar{X}_B\right)^2 \tag{43.5}$$

Finally, the total variability *between* Sample A and Sample B is:

$$SS_A = \sum_i \left(X_{Ai} - \bar{X}_{AB}\right)^2 + SS_B = \sum_i \left(X_{Bi} - \bar{X}_{AB}\right)^2 \tag{43.6}$$

We can now define the variability due to error. This is the total variability *within* Sample A and Sample B—this variability is not due to manipulation of the independent variable and is thus regarded as a source of *error*:

$$SS_{error} = SS_A + SS_B \tag{43.7}$$

The total variability *between* Sample A and B is:

$$SS_{total} = SS_{A+B} \tag{43.8}$$

Finally, The *effect* is the part of the total variability that cannot be explained by the source of error:

$$SS_{effect} = SS_{total} - SS_{error} \tag{43.9}$$

What we have effectively done is we have partitioned the different sources of variability in the samples: (1) variability due to error, that is, the sum of the variability within each group; (2) total variability across both groups; and (3) variability due to an effect of a manipulation of the independent variable—variability that cannot be explained by error. This partitioning is shown graphically in Figure 43.3.

We have found a way to separate out the error from the effect in the data. We first measure the variability of the data within each group (group A and B separately). This gives us the error. Thereafter we measure the total variability (collapsing group A and B into a single group). This gives us the effect + error. We can now obtain the effect by subtracting the error from the total variability.

The sums of squares provide unscaled measures of variability in the data. This can be readily observed because as we keep adding more and more summands the sum becomes larger and larger. Since sums of squares are unscaled they need to be eventually normalized so that we can compare different sums of squares.

Scaled sums of squares are called *mean squares* (MS). Mean squares are obtained by scaling sums of squares by their degrees of freedom (df).

First, we have the degrees of freedom df_{error} within group A and group B, recall this is the error. df_{error} is the number of ways you can arrange the residuals and still have them sum to zero for each group:

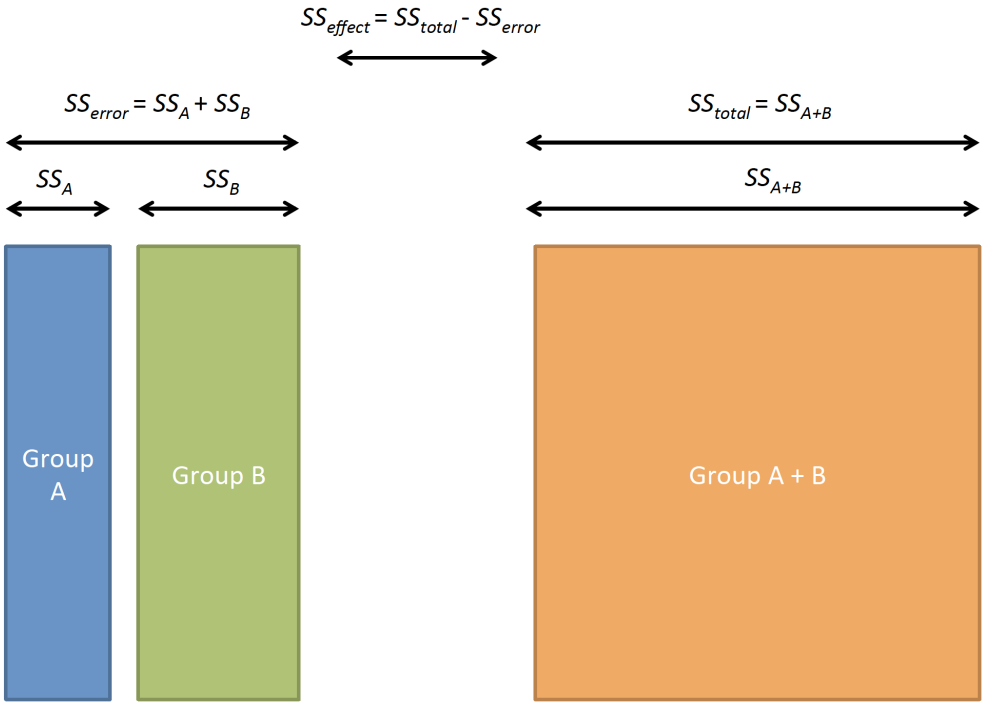


Figure 43.3.: A graphical illustration of how one-way ANOVA partitions sums of squares.

$$df_{error} = n - \text{participants} - m \text{ groups} \tag{43.10}$$

Then, we have the degrees of freedom df_{effect} between group A and B, recall this is the effect. df_{effect} is the number of ways you can arrange their deviations away from the mean so that their average always sum to zero:

$$df_{effect} = m \text{ groups} - 1 \tag{43.11}$$

Now we can obtain the mean squares for error and effect:

$$MS_{error} = \frac{SS_{error}}{df_{error}} \tag{43.12}$$

$$MS_{effect} = \frac{SS_{effect}}{df_{effect}} \tag{43.13}$$

Finally, we can calculate the F-ratio, frequently referred to as the F-statistic:

$$F = \frac{MS_{effect}}{MS_{error}} \tag{43.14}$$

The F-ratio will become large if the effect is larger than the error. Vice versa, if the F-ratio will become small if the effect is smaller than the error. This is because the ratio of the between-groups estimate and the within-groups es/mate gives rise to an F-distribution when H_0 is true.

The F-distribution varies as a function of a pair of degrees of freedom, one for each of the variance estimates.

43.11. Experiments need Explanations

The primary goal of evaluation is to estimate the value a design offers to users. Often this goal is better achieved if the obtained result can be explained.

A small thought experiment illustrates this. You run an evaluative study of a prototype design and find that average task completion time is 47.5 seconds, average error rate 0.5, and SUS (system usability scale) 65. What do such results tell you and how confident can you be when taking further actions based on them?

Explanations help understand distributions of dependent variables. Why, for example, had task completion time an average 47.5 s and not, say, 14.4 s? Data collected during the experiment may offer explanations. Verbal protocols, video recordings, interviews etc. can illuminate *explanatory mechanisms* that link independent and dependent variables. Often in HCI experiments, we discover usability problems, conceptual misunderstandings, or issues in perceptual or motor performance.

Without such explanations, results may be 'fragile'. There is a threat that the obtained results do not generalize. Quantitative findings may be underpinned by factors that are contingent (dependent on) experimental conditions. If those conditions change, even slightly, the result may change, too. For example, if your users are not native speakers,

perhaps they failed to find an item because they missed the meaning. Now, were you to deploy your system to native speakers, the results might change entirely.

We may also seek explanations outside of data – from theories. For example, theories of cognition may help understand why users have hard time recalling facts or events, and theories of communication may expose why users do not want to engage with a user community. For example, Cockburn et al. [159] used decision-making theories from psychology and economics to explain user behavior in using intelligent text entry systems. Because the effect of an intelligent text entry method can be fleetingly small, and contingent on a number of factors, it is important to understand the mechanisms that underpin a user's decision to use a method in a particular way.

To sum up, results of experimental research are more robust and generalizable if they can be explained.

Summary

- Experiments helps give precise measurements and comparisons.
- Explicating research questions and hypotheses is important for high quality evaluations.
- Validity and reliability are key concepts for ensuring trustworthy experiements.
- Data analysis uses methods from descriptive and inferential statistics to draw conclusions about the effects of independent variables on dependent variables.

Exercises

1. Understanding experiments. For each of the questions below, discuss if an experiment is a suitable evaluation approach.
   a) What does it feel like to interact with a chatbot?
   b) Why do people not upgrade software on their devices?
   c) How quickly can people input text on a QWERTY keyboard?
   d) How do we figure out how much people use their mobile phones?
   e) What is the most effective way of organizing email?
2. Formulating testable hypotheses. Please create a hypothesis for an experiment investigating the effects of embodying different avatars in a virtual reality game. Check if you have clear dependent and independent variables. Explain these links with theoretical hypotheses if you can.
3. Experimental design. Please design and run an experiment that will investigate the appropriate mid-air gestures turning on and turning off a television.

4. Comparative studies. You have been asked to run a study that investigates whether a mobile phone application for exercising more during the workday is better than a baseline application. Please consider to think about ”better’ and how to select dependent variables for the study. One of your collaborators suggested using step count; what are your considerations about validity for this dependent variable? Another colleague considers how to capture the subjective of exercising; how do you engage in discussing this consideration?

5. Measuring user experience. Everybody wants satisfying user interfaces. Imagine that you are conducting an experiment where you are interested in satisfaction. Consider how you would operationalize that in the experimental setup that you imagine. What are the essential and less important aspects of satisfaction? How will you measure them?

6. Improving experimental designs. Read the paper by Card et al. [128] and come up with an improved version of the experiment.

7. Reflecting on evaluative practices. Describe how you have previously evaluated designs or software, for example on classes or in your work. Then compare that to the usability testing method as described in this chapter. Give a concrete example and tell how that was tested. Explain whether it was tested from a human-centered perspective. Describe why, or why not, this was done. Next, compare this previous approach to usability testing. Is testing with people relevant or not? Is it your responsibility? What happens if no testing with people is done?

44. Field Evaluations

The evaluation methods considered so far are similar in one important respect: they do not consider the context of use. The methods concern primarily controlled settings (e.g., think-aloud studies) or general models of human performance (e.g., KLM). In the terms of the overview chapter (Chapter 40), those methods maximize precision and generalizability. But what about realism—is not important to get as close as possible to the real contexts of use in evaluation? Collaborative, communicative, and material practices are bound to the contexts in which they emerge.

This poses a formidable challenge for evaluation. If we wish people to engage in their real tasks, in their real contexts, and fueled by their own motivations, how do we carry out an evaluation? If we wish the social and organizational context to remain unaffected by evaluation, how do we study it without changing it? In fact, all the challenges outlined in the User Research part (III) seem to apply here. If we want to study an interactive system in its real setting, we need a fully developed system, and we need people and organizations willing to use the system as part of the evaluation.

This chapter concerns evaluation using field and deployment studies. This encompasses three ideas.

- Field evaluations of prototypes: One idea is to bring out prototypes of interactive systems in their thought use setting; viz., the *field*. As an example, Bardram et al. [41] conducted a 14-week field trial of a personal monitoring system for bipolar patients. The key focus here was on the patients' experiences with the system as part of their lives.

- Pilot studies: Another idea is to partially implement an interactive system and put it in use to learn about its use. Then, those learnings are used to improve the design of the system and fully implement it. For instance, Hertzum et al. [332] studied a pilot implementation of a system for patient transportation at a hospital. The pilot implementation showed a potential of the system to support porters' self organization, but that crucial functionality in the system were missing for that goal.

- Deployment studies: A third idea is to gather information about a fully functional system and improve it as part of its maintenance. DiMicco et al. [193], for instance, built and deployed a social networking system called BeeHive within IBM.

Next, we elaborate on each of the three types of studies. We reflect on the pros and cons of field and deployment studies, including their relation to user research (as described in Part III).

Paper Example 44.0.1 : Using an app to reach AAC professionals in the field

Augmentative and alternative communication (AAC) is a field investigating techniques and approaches for enabling nonspeaking individuals with motor disabilities to communicate. A challenge in such research is to reach users to evaluate prototypes.

Fontana and colleagues [248] presents research that is based on reaching AAC professionals by releasing an app on an app market. The app presents a novel interface for AAC users that rely on symbols to communicate. The app enables the user to select a photo from a photo album. The app will use then use a system to extract meaning from this photo and ultimately generate a set of symbols for the user to choose from. These symbols will be associated with words. In addition, the system will predict phrases, consisting of sequences of symbols. The interface components are shown in the figure below. Users are able to reorder the symbols, remove them, edit words associated with symbols, and add new words and sentences. The user can speak a symbol or phrase by tapping on it.

To evaluate the system, the researchers released the app on an app market and were through the app able to recruit AAC professionals willing to try out the app with AAC users. This enabled the researchers to carry out in-depth interviews with AAC professionals, reporting on the usefulness of the app for their users in-situ. It allowed the researchers to understand how this app can be used to support language learning in a school and in therapy. They discover that the immediate access to a relevant vocabulary by tapping a photo contributes to reducing the conversational partner's workload, and may help AAC users in understanding symbols and sentence construction. Overall, the paper reports a rich interview material from a diverse set of AAC professionals which would be difficult to source using traditional means.

44.1. Field Evaluations

One major decision point for evaluations is whether to carry it out in the laboratory or in the field. In a laboratory evaluation, an evaluator can control what happens during an evaluation: the tasks users are carrying out, the instructions they receive, and other factors that may affect user performance and behavior. Researchers can also ensure that observations are independent, that is, that participants do not influence each other. These practices minimize the risk that factors irrelevant to the aim of the evaluation compromise the validity of any conclusions drawn from a study.

Field evaluations are attractive because they tend to provide a higher *ecological validity*: The conditions in which the study is arranged have a higher resemblance to the real-world conditions of use that a system might be used in.

44.1.1. Degrees of Field

It is convenient to think about the field as a matter of degree. So, rather than a strict distinction between lab and field, we can imbue evaluations with more or less features of

the field. Again, this is about realism, in the sense of [518]. So evaluations may be more field-like in terms of people, activities, contexts, and technologies (see Chapter 10).

Let us give some examples. For instance, many think aloud studies can be done at users' workplace, ensuring that the physical and social contexts are more similar to the imagined use context than a usability laboratory. This is similar to the approach by Rico and Brewster [679]. They evaluated the social acceptability of a gesture-interaction set for mobile phones on the pavement near a bus stop and an underground station on a busy city street. In that way, the context of the evaluation was allowed to influence its results and increase its realism. We can also vary the activities that people engage it, for instance the tasks. This is why tasks in think-aloud studies are sometimes supplied by users themselves (see Chapter 42). This make them more realistic and closer to what would happen in the field (aka real life). In the study by Rico and Brewster [679], although the setting was the field, the researchers gave the tasks.

In an attempt to get evaluations to be more like the field, researchers sometimes focuses on particular aspects of the lab–field continuum. In the 1980s and 1990s, usability researchers were spending energy on constructing usability labs that looked like users' workplaces or experimental settings constructed to resemble living rooms. Such attention to *mundane realism* is warranted but is likely less important than, for instance, ensuring that participants in an evaluation sees it as meaningful and engaging or that the tasks in an evaluation reflects tasks participants would do in real life [351].

44.1.2. Evaluation Methods for the Field

If we wish to increase the realism of an evaluation and do it in the field, which evaluation methods can we apply? The short answer is that all empirical evaluation methods may be used. Thus, we can use all the method covered in this part, including the think aloud method in the field and experiments.

However, we can also use the methods discussed in the part of user research III, including interviews (Chapter 11), observations (Chapter 12), and unobtrusive data collection (**??**). Recall that the goal of an evaluation is to valuate a system against some standard. In contrast, the studies discussed in the part on user research were often about understanding users existing work. Thus, the intention differs between the use of, say, interviews for user research and for evaluation.

44.1.3. Experiments in the Field

One method that has been particularly adapted to the field is the experiment. Field experiments need to balance the causal logic of experiments with the realism of the field. As an example of how the deployment environment can change results, consider a field test of two deployed mobile text entry methods: an auto-correct keyboard and a gesture keyboard Reyal et al. [678]. This field test compared the lab text entry performance of both text entry methods with the text entry performance that could be measured when users were using both text entry methods installed on their phones and used in everyday life.

When thinking about experiments in the field, three core assumptions of experimental research are threatened: random assignment, control, and independence of observations. First, we may not be able to randomly assign users to particular conditions in the field. We need to work with users who happen to use some user interface or have a need to carry out particular tasks.

Second, we cannot control events to the same extent as in a laboratory setting. Field studies are open to a surprising number of unexpected events and factors that may substantially affect any outcome. For example, while you may ask users to carry out their everyday work task using a prototype instead of some established user interface, something unexpected may happen, such as a crisis at work, new work rules, a new workflow, or new colleagues, that alter the way users interact with the prototype.

Third, our observations may not be independent of each other. For example, if you want to evaluate a system with a school class, it can—or, rather, will—happen that students talk about the experiment with each other, which will influence their behavior.

In the literature on experimental methods, field experiments are sometimes called *quasi-experiments* for the reason that they often violate assumptions of experimental research [739]. Quasi-experiments call for caution in analyzing data. We cannot rely on the same statistical methods for drawing conclusions from data collected in field experiments.

A particular type of quasi-experiments are natural experiments. In such experiments, a happening in the world works as the experimental manipulation—the influence of COVID-19 on communication is one of the most well-known natural experiments. Such a happening allows us to study its influence in the real world. It is not a full experiment because assignment is not random and the happening may be associated with many causes.

As an example, Griggio et al. [287] studied the influence of a new privacy policy for WhatsApp, in particular if the policy would make new users switch to other platforms. Using surveys immediately after and a while after, the authors were able to conduct an experiment-like comparison using an happening in the real world.

Paper Example 44.1.1 : Field evaluation of an interactive tutorial for Wikipedia

Sometimes evaluation studies can only be carried out in the field. Narayan et al. presented an interactive tutorial for learning Wikipedia [561]. The goal was to encourage positive behavior and high quality contributions. Wikipedia suffers from high rate of change in its communities, and newcomers should be quickly mobilized to 'the ways of the house'. It is also vulnerable to norm violations and malevolent behavior like trolling. Such behaviors can compound, as newcomers are less likely to join communities where they experience demoralizing behavior by others or are not assured by the quality of contributions. TWA – The Wikipedia Adventure – is structured in the form of a game (adventure) and uses methods from gamification research (see Chapter 8). It covers critical ares affecting Wikipedia contributions: wiki-markup, community policies, and communication with other editors. It allows exploring and learning in a 'sandbox', safely without the fear of 'looking stupid' among other editors. Users go through seven 'missions' [p.1788]: ''setting up their user page, communicating effectively with other users, making basic edits, maintaining a neutral point of view, evaluating content quality, understanding revisions, and using built-in tools like watchlists and history pages to see how articles can be maintained over time." Missions are themed as stories and completing them is rewarded with 'badges', a common gamification technique.

The authors hypothesized that, in order to become a Wikipedian, novices should be able to perceive that even tiny contributions can be valued in the community, when done according to the norms of the community. To arrange a field evaluation, the authors collaborated with Wikimedia Foundation. They first organized a user survey among Wikipedia editors, who were invited to comment the tutorial. The goal was to collect *formative* feedback that helped the authors evolve the design of TWA. 90% of 600 respondents reported feeling more confident as an editor after passing the tutorial and helping them to understand Wikipedia better. However, the authors noted that a survey-based evaluation with experienced editors may not carry over to newcomers, which was their initial target group. In the second phase, they wanted to see if TWA would *actually* help newcomers to increase the quality of their contributions even after completing TWA. They hypothesized that gamification might help them increase confidence and self-efficacy (beliefs that they are able to master Wikipedia). To recruit users, the team invited users via talk pages to play TWA. This, obviously, limited their sample to newcomers who chose to opt-in to the study. To learn about the *effect* of TWA, the authors needed to compare two types of newcomers: those using and those not using the system. Their solution was to divide invited users according to whether they chose to use TWA or not. The authors placed opted-out users to a control group.

Contrary to results from Phase 1, the authors found no measurable effect of TWA on newcomer participation in Phase 2. All outcome metrics the authors used, which measured the quality of contributions 180 days after taking TWA, failed to reject the null hypothesis of no effect. The authors attributed this to the self-selection effect: that that users who chose to take TWA were different from others. The authors concluded their evaluation: ''In a project like Wikipedia that depends heavily on intrinsically motivated members to make contributions, a gamified tutorial may be helpful and fun to use, but ultimately unsuccessful at building long-term commitment and retention" [p.1796].

44.2. Pilot Studies

In item VI we discussed prototypes as a way to get hands-on experience with a design. Can we do that in the field for systems that have been sufficiently developed to be used there? If so, this would allow us to evaluate a system in its intended use context.

Pilot implementations have emerged as an answer to this question [331]. A pilot implementation is "a field test of a properly engineered, yet unfinished system in its intended environment, using real data, and aiming—through real-use experience—to explore the value of the system, improve or assess its design, and reduce implementation risk". This allows evaluators to test the entire sociotechnical system (see Chapter 33) rather than a smaller part of the interactive system.

A pilot implementation differs from a prototype in a few, key ways: (1) it is always done in the field, (2) it is "used in its intended environment for a limited period of time, with real data and special precautions against breakdowns" [331], (3) it is done when feasible to test both the design, the engineering, and the implementation, and (4) it is done for weeks or months. In these ways, a pilot implementation is a partial deployment study.

Let us consider an example where a pilot implementation was used. Systems in health care are complex, but in particular their fit to the organization at hospitals have proven surprising when many systems have been brought into use. As part of the development of an electronic patient record (EPR), a part of the EPR was pilot implemented at a hospital [329]. For five days, the pilot implementation of the EPR replaced all paper records in the hospital unit. A back office was staffed for all five days to help resolve breakdowns, unavailable features, and other contingencies.

The pilot implementation showed that the EPR and the associated work procedures were successful in achieving many of the changes that were planned. Some consequences of the interactive systems that were unexpected were also found. For instance,

> the nurses engaged in a process of collective reading at their handovers, during which the EPR screen was projected on the wall and thereby visible to everybody. The electronic records were inspected by the group of nurses, and they collectively participated in interpreting the status and condition of the patients, guided by the nurse team leader. The nurse team leader navigated the EPR and read selected passages aloud to draw attention to them and set a shared flow in their reading. This collective reading was a marked change in the nurses' work practice. During nursing handovers with paper records the nurse team leader provided an oral report of each patient by scanning the patient's record and reading key information out loud; patient records were seldom seen by clinicians other than the nurse team leader.

In this way, the pilot implementation helped the evaluators uncover emerging but important phenomena surrounding the use of the interactive system.

Pilot implementations as a concept are related to other ideas about bringing essential parts of an interactive system into use in the field to be evaluated. With *technology probes*,

researchers deploy simple and adaptable interactive systems [359]. The goals of doing so are threefold.

- to understand the real-world needs and desires of users,
- to help users and designers conceptualize new technologies, and
- to field test the interactive system.

In that way, technology probes bridge work on understanding users, on designing interactive systems, and on testing the interactive system in the field.

Another related concept is *the minimum viable product* [681]. The idea is to create and release a version of a new interactive system that allows its designers to collect the maximum about of information from users with the least information. That way, actual behavior with the product may be observed and use to improve it and take business decisions about its future. As with the general idea of pilot implementation, the product is partial and need not be fully working but perhaps just simulated behind the facade of an interactive system.

44.3. Deployment Studies

When a product, system, or service has been designed and built, it is ready to be *deployed* to end-users. Often ignored in research, deployment is a critical stage in the development of any product, system or service. In deployment studies, interactive systems are fully deployed and as they are used, evaluators collect data about the success of the system.

The reasons for conducting such evaluations vary. First, support costs might be reduced by learning from the issues that users face with the real-life use of a product. Second, evaluators may be interested in ensuring that the intended effects of a system are realized; studying the deployment of systems is a way of doing that. At a fundamental level, it is only at the deployment stage that it will be possible to fully know that a system is fulfilling its purpose in helping users achieving their goals in relevant use contexts. Third, future versions of the system may be improved through the feedback. Deployment allows studying systems released to the actual intended users who will use these systems to achieve their goals in their sociotechnical context.

Chilana et al. [149] conducted a large-scale survey of what they call post-deployment usability. They showed that only half of the 333 HCI professionals are involved in usability work after the deployment of a system. Frequent activities for studying deployment include interviews/surveys, informal usability testing, analysis of customer support data, satisfaction surveys, and monitoring of discussion forums, and analysis of logfile data.

Let us next look more closely at a few examples of methods used in deployment studies. As noted earlier, many of those methods were covered already in the part on user research (Part III).

44.3.1. Analysis of User Feedback

One form of deployment studies are to gather users' feedback and use that to evaluate a system. Of course, such feedback may be elicited through questionnaires (see Chapter 13) or interviews (see Chapter 11). Howevver, a prominent approach is to use feedback that users give independently of the evaluation. Users may provide such feedback in the form of support calls, reviews of the system, or comments on community support forums. For example, Zhai et al. [904] deployed the ShapeWriter WritingPad on an app store and analyzed reviews that users wrote about the app. All of these types of feedback may be analyzed to evaluate the deployed system.

44.3.2. Logfile Analysis

Similarly to unobtrusive research (see Chapter 14), evaluators may use web site analytics and logfile analysis to evaluate a deployed system.

44.3.3. App Store Deployment

One of the main costs in deployment studies are distributing the software to users. A popular methodology in HCI research was developed along with the rise of app stores for downloading software for mobile phones and tables [520].

44.3.4. Longitudinal Studies

Longitudinal evaluations are field evaluations that follow a prototype or product for a longer period of time. Although such evaluations may be done in the lab, they often draw on the intrinsic motivation of field evaluations.

So what counts as longitudinal? There is no clear answer, however, longitudinal studies typically have a duration of weeks, months, or even years. The duration of study depend on what is examined, for example, habits and social practices may require longitudinal studies of a very long duration.

Longitudinal studies are not only expensive to organize, but also difficult to analyze. During an extended period of time, numerous events can occur that change the subject of evaluation. Users may not only develop as users but also go through significant life events, such as changing jobs, getting children, and so on, that may change the way they use the system. However, we sometimes wish to carry out evaluation studies in the field. Such studies can be organized by, for example, recruiting users or groups to adopt a prototype for some agreed-on period of time. Modern software markets and app stores also make it possible to deploy prototypes to end-users directly.

For example, Vitale et al. [838] observed users during an operating system upgrade and followed them over four weeks afterwards using a diary method. An often overlooked source of frustration for end-users are the updates and upgrades needed for an interactive system. Yet, the ways that updates and upgrades happen are designed and those designs may be evaluated for their effects on users. Their field data showed that participants had negative reactions to the upgrade process.

44.4. Is it worth the Hassle?

Field studies offer the highest realism out of evaluation methods in HCI, however they are also costly. Are field studies "worth the hassle"? Researchers most of the time decide "no" [414]. At the same time, many argue that field-based evaluations are critical, even if they costly and difficult to conduct. What are such insights that can be obtained only in the field?

A study looking at the usability of a mobile medical informatics system tried to answer this question [414]. To the surprise of the researchers, a comparable amount of usability problems were found in both settings. Moreover, usability problems that were closely intertwined with the use context were found both in the lab and in the field; finding such problems would typically be considered a benefit of field-based evaluations only. So in this particular study, the field-based evaluation were not worth the hassle.

We find these views problematic. Field and deployment studies have a number of pros and cons. In line with earlier methodological discussions in this book, neither field studies or lab studies are inherently good. Rather, they are methodologies that allow us to focus on certain information with associated methodological qualities (see Chapter 10). Table 44.1 contains a summary of the main tradeoffs to consider when choosing between field- and lab-based evaluations. Field studies are stronger in realism, which allows them to expose phenomena that laboratory studies might not reach. On the other hand, they lack the precision and cost-efficiency that lab studies offer.

Summary

- Field and deployment studies help evaluate systems by enhancing the realism of the evaluation.
- Realism and therefore the field is a matter of degree; many evaluations may be improved by increasing their degree of realism.
- Systems may be partially or fully deployed to evaluate them.

Exercises

1. Methods. Recall the field evaluation methods discussed in this chapter. Which method woudl you use for the following research needs: 1) learning how Facebook users use a new AI-boosted newsfeed; 2) learning about the usability of a tangible UI design for aging adults; 3) learning about the navigation behavior of users on a newly launched banking web site.
2. Experimental control in the field vs. the lab. Consider having to run a usability evaluation of a wearable interface developed for firefighters that enables them to communicate without a mobile device when on a mission. How would you arrange a 1) laboratory-based usability study and 2) a field-based usability study? Discuss

Type	Pro	Explanation
Field		
	Realism	Brings evaluation to the context and activities in which the interactive system will be used.
	Socio-technical fit	Enables the discovery of the fit, or lack thereof, between social activities and structures and the system.
	Can be of long duration	Field evaluations typically last from a few days to a few months
Lab		
	Precision	It is easy to obtain detailed measures of the use of the system and to control other factors that are not being studied.
	Low-cost	Requires fewer resources than a field study.
	Can evaluate unusual tasks	Tasks that occur rarely 'in the wild' can be arranged
	Do not require a robust system	Studies can be run with rough prototypes

Table 44.1.: An overview of the main differences between doing evaluation in the field and in the lab. Note that this is phrased as a binary decision, although it is a matter of degree (see the text).

external validity and construct validity of each. Which aspects can be appropriately staged in laboratory setting and which cannot?

M