

[T32]

Usability by Inspection: Collaborative Techniques for Software and Web Applications


Lucy Lockwood and Larry Constantine

Abstract

Usability inspections are a cost-effective alternative to conventional usability testing for evaluating and improving the usability of software and Web-based systems. The collaborative usability inspection technique described here is a highly structured, scenario-driven inspection process optimized for rapidly identifying, classifying, and prioritizing usability defects. The process has been refined through extensive application into a highly efficient, easily learned technique. The basic structure is introduced and guidelines for conducting successful inspections are described.

Introduction

Even the best design has flaws, and even the best designer makes mistakes. Since there will inevitably be usability problems, how do you find them while there is still time to fix them? Once a product has shipped or once a Web site goes live, usability problems can have costly consequences. Tech support and training requirements can become overwhelming, sales can be lost, expensive reworks and field fixes may have to be deployed, and customers can become alienated. The goal in product development should always be to find problems as early as



possible, when fixing them is a matter of changing a few pencil lines rather than rewriting code, when corrections can be made quietly and without fanfare.


Fortunately, to help designers locate and fix usability problems a variety of techniques are available, including expert evaluation, usability testing in the field or in a usability lab, and usability inspection processes. Expert evaluation—particularly the largely standardized Heuristic Evaluation technique (Nielsen, 1994)—and usability testing (Rubin, 1994) are the mainstays of modern practice among usability professionals. However, both testing and expert evaluations have their limitations.

Expert evaluations require experts and may be difficult for beginners to learn to conduct effectively (Chattratchart et al, 2003). As typically performed, expert evaluations may not effectively reflect the perspective of real or representative users or the tasks they need to perform (Seiden, 2003).

Usability testing, long the centerpiece of user-centered approaches to assessing usability, is a broadly useful technique but also has its limitations and shortcomings. Genuine testing requires something to test—a working system or prototype or at least an effective simulation—which means that testing involves a significant up-front investment before it can be applied and tends to concentrate its results toward the later stages of the overall product development cycle. Although some writers advocate so-called paper prototype testing, strictly speaking, what they describe is a form of inspection. An electrical system in a building can be tested, but one cannot test a blueprint or a drawing of the building. Code can be tested, a drawing of a screen can only be inspected.

Although simplified equipment and streamlined test protocols based on small samples of user subjects can reduce the cost of usability testing, in practice, usability testing is not extremely efficient in terms of the hours of testing and analysis required to uncover a given number of problems. More significantly, although testing can be effective for identifying significant but localized design errors, it is relatively less effective at exposing basic flaws in the architecture or overall organization of a user interface.

Inspection techniques, such as code reviews and design walkthroughs, have a long history of success in uncovering defects in software. In an inspection, software or a description of software is reviewed systematically with the objective of finding bugs, errors, problems, or flaws. The principle advantage of inspections over testing is that almost any artifact at nearly



any stage in the development process can be inspected, while only executable code can be tested. Thus, inspections can provide feedback about potential problems or shortcomings much earlier in the process, at a point when corrections are simpler, more straightforward, and less expensive.

Collaborative Usability Inspection

Usability inspections are a method of quickly and efficiently identifying usability defects in software, Web sites, Web-based applications, or even hardware equipment. Inspections can be conducted on a variety of design and development artifacts or work products. They can be applied to design models, such as navigation maps or abstract prototypes (Constantine, 1998; 2003), to paper prototypes or user interface mockups, to working prototypes or simulations, or to completed systems at any stage in development. Repeating inspections throughout successive stages in the development cycle can further increase the ultimate usability of a system.

Collaborative Usability Inspections (CUIs) are a specialized technique originally invented by the authors (Constantine, 1994; Constantine and Lockwood, 1999) to support ease of learning and application and to enable developers and other professionals with little or no background in usability or user interface design to achieve immediate and substantial improvements in usability.

CUIs owe their efficiency and effectiveness to a highly organized process with a number of distinctive features worked out through a decade of use in a wide variety of settings. These include:

- a sharp focus and narrowly defined purpose
- assigned and carefully delineated roles for participants
- an orderly, well structured process
- scenario-driven, principle-based inspection procedures
- systematized rules of conduct
- collaboration among designers, developers, usability specialists, and users

The result is a remarkably efficient technique that can be quickly mastered even by relatively inexperienced teams. With practice, inspection teams can often uncover as many as 100 defects per hour of inspection.

Usability defects.

Like the inspection process in a factory, the purpose of a Collaborative Usability Inspection is to find defects—as many as possible and as efficiently as possible. It is *not* about congratulating designers or developers on their good work. It is *not* for designing, discussing, or debating. An inspection that finds few defects is regarded as a sign of failure of the process not success of the product.

To encourage the aggressive search for defects, CUI participants are reminded that just because a defect has been identified does not mean that it must be or will be fixed. Just as the factory inspector is only expected to find the defective units not to repair them, so the CUI team identifies usability problems but does not design solutions. Redesign is a separate activity to be undertaken at another time if at all. The choice of whether or not to correct usability problems is a business decision that depends on priorities, resources, and business goals for the product. The underlying philosophy is that it is better to know where the problems are in order to be able to make a deliberate decision than to be at the mercy of unknown defects.

The CUI technique hinges on a clearly defined notion of what constitutes a defect. A usability defect is defined as

any potential problem in the operation, appearance, or organization of a system that makes the final product less easily used by its targeted population of end users.

Operationally, a usability defect is either

- a clear and evident violation of or departure from accepted usability principles or
- a probable cause of delay, confusion, or error on the part of a user or a contributor to the user's failure to complete an intended task.

Usability defects are thus defined either with respect to an agreed upon set of principles or by reference to user performance. Any agreed upon collection of design or usability principles may be referenced in a CUI. The so-called Principles of Good Form (Constantine and Lockwood, 1999) that we most commonly employ are summarized in Table 1. Nielsen's Heuristics (Nielsen, 1994), summarized in Table 2, are also widely used. Other principles that might be referenced in the course of some inspections include such things as accessibility or universal design principles.

Table 1 – Principles of Good Form

Principle	Description
VISIBILITY	Everything to complete a task should be available and apparent when and where needed.
FEEDBACK	Users should be kept informed of consequences of actions, events, and progress as relevant to them and their tasks.
STRUCTURE	Layout should be organized by meaning and use and should makes sense to users in terms of their intentions.
REUSE	Purposeful consistency should be maintained through reuse of internal and interface components and behaviors.
TOLERANCE	The user interface should provide flexible and forgiving organization and interaction.
SIMPLICITY	Important and frequent tasks should be simple and straightforward.

Table 2 – Usability heuristics (from Constantine and Constantine, 1999: 64; based on Nielsen, 1994)

Heuristic
Visibility of system status: keeping the user informed.
Match between system and real world: user language and real-world conventions.
User control and freedom: easy exits, undo, and redo.
Consistency and standards: following them appropriately.
Error prevention: reducing user mistakes.
Recognition rather than recall: reduced need for remembering through visible options, actions, and instructions.
Flexibility and efficiency of use: customization and support for advanced users.
Aesthetic and minimalist design: reducing irrelevant or rarely needed information.
Error handling: helping user recognize, diagnose, and recover.
Help and documentation: comprehensive and accessible.

In all cases, for some element of a user interface to be considered a usability defect, it must have a demonstrable or arguable impact on user performance or it must violate or depart from agreed upon usability principles. It is not something that someone merely dislikes or wants to change. If an inspection participant who identifies something as a defect cannot show cause by citing a principle or by relating it to an actual problem in user performance (error, confusion, delay, or the like), then it is not considered to be a defect. In this way, the subjective aspects of evaluation are significantly reduced even if not completely eliminated.

Usability defects are characterized by—and recorded in terms of—five elements: what, where, how, why, and how bad.

1. What: the particular feature, function, or facility that is the problem.
2. Where: the exact location of the defect within the user interface.
3. How: description or nature of the defect.
4. Why: the particular rationale or principle that makes it a defect.
5. How bad: the estimated severity of the problem in the context of how much it impairs a particular task and the application as a whole.

For example, a usability defect might be described as follow:

- (1) Command buttons for “concatenate” and “query” in the
- (2) “Customer Data, Advanced Operation” dialogue (3) is confusing because (4) it uses programming rather user terms, (5) a problem of moderate severity.

Defect severity.

Although almost any scale could be used, we find that a four-point scale ranging from “nominal” through “critical” suffices for most purposes. The levels in this scale are:

1. Nominal: A nuisance or annoyance, a contributor to insignificant or infrequent delay, or a small chance of user error.
2. Minor: Some chance of impairing user performance, impeding learning, or increasing user errors somewhat.
3. Major: Some tasks will be substantially more difficult to perform or master or the likelihood of error is significantly increased.

4. Critical: A blatant defect, significantly reducing overall usability or making the product substantially unusable for some purpose or purposes.

The severity of a defect is always contextual; it depends on the application and the specific user tasks that are affected. For example, an input field that rejects non-numeric characters might be a major or critical defect if it is used for credit card input on a consumer-oriented Web site, since it can lead to customers bailing out during the checkout process. (Nearly all credit card numbers are embossed with spaces between groups of digits, and users are apt to type them that way. Requiring them to be entered in a different format increases the probability of a typo and makes it more difficult to visually verify the number against the card.)

Usage scenarios.

One advantage of CUIs is that they employ a scenario-driven inspection process. Inspection scenarios provide a framework within which to inspect the various parts of the user interface. They take the user or users and other inspection participants through various parts of the user interface in the course of acting out a plausible interaction with the software application or Web site. To a considerable degree, the success of the inspection in finding defects will depend on the quality of the inspection scenarios. To be effective, inspection scenarios must be neither too simple nor too complex. If the scenarios are too easy or straightforward, they will not sufficiently exercise the user interface; if they are too involved or complicated, the inspection will get bogged down dealing with problems in the scenarios at the expense of finding problems in the user interface.

Good inspection scenarios are based on a task model that embodies the designers' understanding of the work and intentions of users. In usage-centered design, the task model takes the form of a collection of individual task cases (essential use cases), but other forms of task models can be used to guide construction of inspection scenarios.

The aim is to significantly exercise the user interface by incorporating a range of tasks or task elements. These include representative or typical tasks and common or frequent tasks as well as key or critical tasks and important exceptions or special cases, whether or not these are common, along with an assortment of other tasks that are of interest for one reason or another.

To construct an inspection scenario, these individual tasks must be combined into groups that can form the basis for a plausible, believable

storyline that would make sense to a real user. For example, consider a usability inspection of a remote control for a VCR. The task model might include the following task cases, among others:

advancing to general section
backing to general section
picking exact spot
adjusting picture to improve quality
recording future show
adjusting clock


Here are two examples of brief inspection scenarios that might be constructed based on these tasks:

“You are a movie buff looking forward to studying a classic film. You want to play a particular scene on the worn video of that much-loved and much-viewed movie. Try to get a stable picture by adjusting the tracking, then find the exact start of the scene you want and play the first few minutes of it twice.”

“You are leaving on a trip and remember at the last minute before leaving for the airport that there is a PBS special you want to record. It’s Sunday now, and the special runs Tuesday through Thursday from 10-11:30pm. After you complete the programming, you notice that you had forgotten to reset the clock on the VCR to Standard Time, so you adjust the clock, then check to verify that your programming is okay.”

As in the condensed examples above, the inspection narrative should address the user directly and should begin by supplying the context and motivation to the user. For example:

“You are a few minutes late starting your first shift as a telephone order taker on the phone bank. Yesterday you were given a whirlwind run-through of the system, but you remember none of it. The phone is ringing and a caller ID has popped onto your screen. You want to impress your boss, so you decide to just plunge in and try to handle the call.”



Inspection scenarios should be written entirely in the ordinary language of the users and of the application domain. For obvious reasons, the scenario should be expressed in terms of a sequence of user goals or intentions rather than actual steps or actions. For example, “You want to buy a new sweater for a toddler. See if you can find one for less than \$20.” Not, “Click on the Toddler Wear’ link, then open the drop-down to select ‘Up to \$20,’ then click the ‘Search’ button.”

We find it useful to prepare one more scenario than we expect to get through in an inspection—just in case we finish faster than expected. Typically this means two or three scenarios, although it depends on how protracted the scenarios are, how good the inspection team is, and how bad the user interface is. In general, workable inspection scenarios will fit on a single page of paper.

The main points in constructing inspection scenarios are that scenarios should:

- Address user participants directly in the second person.
- Provide motivation and context to facilitate users getting into role.
- Use ordinary language and the language of the application domain.
- Organize the overall process into a series of interrelated steps of manageable size.
- Express steps in terms of general goals or intentions, not specific actions.
- Incorporate a variety of task elements, including exceptional or specialized cases of interest.
- Form a plausible and connected storyline that makes sense to the user.
- Exercise the complete user interface design as thoroughly as practical within the scope and limitations of the inspection.

Each inspection scenario has three main parts:

1. Overview or summary
2. Context and user motivation
3. Sequence

The scenario should begin with an overview or summary of what the scenario is about. The work setting and work-flow context within which the scenario is to be assumed to be carried out is described, along with the users’ motivation. The context and motivation are provided to help the inspection team, and particularly the person who is the user on the team,

to get into the mindset of the role appropriate to enacting the scenario. The scenario sequence presents the scenario as a succession of discrete objectives.


Table 3 below presents a complete example of an inspection scenario.

Table 3 – Example of an inspection scenario for a telephone ticket sales application.

Scenario
<p>Overview:</p> <p>This inspection scenario incorporates both customer inquiry and ticket sales activities for telephone staff in Telephone-Query-Handling Role and Telephone-Ticket-Selling Role.</p> <p>Context:</p> <p>You have just come on duty for your first session as a telephone volunteer when the telephone rings. The caller is interested in tickets for the next concert in the Sunday afternoon Meet-the-Composer series 2 weeks from now, at which she has heard there is a new piece premiering by New Zealand composer Michael Nyman.</p> <p>Sequence:</p> <ol style="list-style-type: none">1. You try to check for such a concert on upcoming Sundays.2. You try to find any upcoming performance of a work by Michael Nyman.3. You inform the caller of the performance(s) you find. When the caller asks for more information, you try to get program details.4. The caller expresses interest in one of the programs and asks for six seats together in the center section of either the main floor or first balcony, not too far in from the aisle and not too far back, but not right up front. You try to check the seating for the concert.5. As there are no six seats together, you try to find two or three suitable combinations of seats and describe them to the caller.6. The caller then decides to purchase one of the seating combinations offered. You get the caller's name and credit card details.7. The caller asks for the tickets to be mailed, so you get the mailing address and complete the sale.

The Inspection Process

Part of what makes the CUI process so effective in finding usability problems is that the interaction is deliberately slowed down to enable thoughtful and thorough examination. The process is driven by a scenario, which guides the exploration of the user interface step-by step. At each step, the user participants are asked to describe what they would do next,



such as open a drop-down, click on a link to a particular page, or the like. Whenever an interaction context is encountered for the first time, the inspection is interrupted to identify evident defects based on first impressions. In this way, every interaction context that is visited gets a thorough review, not just with respect to the elements incorporated in the scenario. At the end of the session, any interaction contexts that were not visited in the course of enacting scenarios are reviewed out of context to make sure as much of the user interface as possible has been seen at least once.

It is best to start the inspection process at the very beginning, with application launch, user log-in, entering the home page URL, or the like. At some point prior to release, some inspection should include installation, setup, or system configuration processes as relevant.

Scope and schedule.


A CUI is a time-boxed process. In our experience, the practical minimum for a productive session is one hour. About two or two-and-a-half hours is ideal, with a brief break at the halfway point. Beyond three or four hours, fatigue sets in rapidly and markedly reduces effectiveness. We have on occasion been forced to schedule back-to-back sessions before and after a midday, break but this can be brutal. As a rule, morning sessions seem to work slightly better than afternoon ones.

Very large and complex systems will generally need to be inspected in sections over multiple sessions. The portion of a user interface to be inspected should comprise a coherent collection of capabilities. Since large projects are often designed and implemented in stages, a workable practice is to inspect each part of the design as it is completed. It is also useful to conduct at least one inspection that reviews the user interface as a whole, with special attention to how everything hangs together and how easy or hard it is to make sense of and navigate through the user interface as a whole.

Enacting scenarios.

Typically, an inspection scenario is introduced by reading the overview, context, and user motivation to the entire group. The body of the scenario should be presented only one step at a time to discourage participants from getting ahead of the process.

User participants are encouraged to “think aloud” about how they see the user interface and the reasoning behind their actions at each step in



the inspection scenario. User participants are always asked to speak first, before any other participants, both in reacting to a new screen or other interaction context and in describing the action or actions they would take at each step in the scenario. In this way the user input is obtained without contamination by comments from designers or developers.


If there is more than one user participant, each is polled in turn about what he or she would do next. The Lead Reviewer running the session has the option of following one particular thread or trying each proposed action in turn, depending on available time and relevance to the inspection goals.

How scenarios are best enacted depends on whether the system, being inspected is passive—such as a paper prototype or mockup—or active—such as a simulation or active prototype, a live Web site, or a software beta release.

With a paper prototype or other passive inspection object, one of the developers or designers familiar with the product should be assigned the role of Simulator with the responsibility of “playing computer,” that is, acting out or describing system behavior as if it were an actual working application. For example, if the user says, “I would click on that icon,” then the Simulator might say, “Okay, then the system would put up a message,” (places a blank message box on top of the screen mockup) “which says something like, ‘Prescription details are incomplete.’”

When inspecting an active system, it is tempting to allow the user to interact directly with the system as would be done in usability testing. However, this can defeat the purpose of slowing down interaction to allow an orderly and controlled inspection of the user interface. It is best to have one of the designers or developers take on the role of “driving” the system under the direction of the Lead Reviewer. After users state what they would do next and the thinking behind their actions, the Lead Reviewer would direct the Driver to carry out some action. It is essential that the driver not take independent action or get ahead of the inspection process, as this can lead to missed opportunity and lost information. For example, if a new screen is revealed prematurely, a user may be tipped off as to the correct action to take so that problems with ambiguity in the interface might not be uncovered.

Throughout the inspection, it is important that participants do not guide or prompt users in any way. Side comments or separate discussions are not only distracting, but can provide cues to users. Even non-verbal behavior, like sighs or sidelong glances, can subtly steer users in one direction or another. In one inspection of a live Web site, for example, the



Driver repeatedly would begin to move the mouse pointer toward a target object before the users had suggested an action. The Lead Reviewer eventually had to replace the Driver to stop this subtle leading of the users.

Help should be available only on request and never offered unasked. A request for help is an opportunity to learn more about problems in the design and about potential solutions. Such requests should be handled as if they were attempts to use the online help system. (“Okay, what would you do with the help system to try and get information on what to do next?”) The help that is given should be limited to the kind of simple description or clarification that would actually be available through the help system. (“So, you clicked on the question-mark tool, then placed the mouse pointer over the check box. A balloon would pop and say something like ‘Check box to show that patient has read and initialed HIPAA statement.’”)

Any request for help by a user is taken as an indication of one or more usability defects, since the standard of performance is that users should be able to complete their work without help. The Lead Reviewer and Recorder decide how such an implied defect should be recorded.

Exceptions, errors, and alternatives should be dealt with as they arise. Should an unanticipated action from users take the inspection into a part of the application that has not been designed or built, the Lead Reviewer can decide whether to pursue that thread or abandon it. (“We have not designed that subsystem yet, so let’s go back to the previous screen.”) If the thread is followed, ad hoc interaction contexts may need to be created. (“Okay, what you would see would be something like the last screen but with all the non-paying attendees highlighted in yellow or something like that.”)

In the event that users become hopelessly lost or are completely unable to proceed with a scenario, the Lead Reviewer may intervene in order to continue the inspection and gain some value. (“All right, let’s go all the way back to the first screen and start over with a different tack. What else might you have tried?”) If the problem was not one of navigation but a show-stopping usability flaw, the Lead Reviewer may want to simply move on to the next scenario.


Inspection Participants

CUIs work best with from 5 to 12 participants who can bring a variety of perspectives to the inspection. In general, as with any inspection process, more pairs of eyes is better. However, there are diminishing returns beyond a certain point. We have conducted a few inspections with more than 20 participants, but too many people can slow the process, and the extra people may contribute little. On the other end of the scale, too few people is also a problem. Inspections with only 3 or 4 require doubling up on roles. Below that threshold, you are probably better off using some other evaluation technique. However, except for users, it is not necessary that participants have some special skills or a particular relationship to the system being inspected. It is quite acceptable to grab people from the office next door to fill out an inspection.

Whenever possible, the inspection should include the designers, information architects, graphic artists, and others responsible for the design of the system being inspected. This educates designers about user experience and usability problems and helps them learn how to avoid many kinds of problems in the future. It also increases the credibility of results and reduces the potential for later disagreement because designers were there to see first hand the nature of the problems.

We also like to include some developers in every inspection, particularly user interface programmers. They, too, learn more about usability and how to spot problems, becoming better able to make sound on-the-fly decisions during software engineering and construction. In general, every inspection becomes an opportunity to disperse knowledge and build skills more widely. For this reason, we also like to include designers and developers from other projects or other parts of the same project. Such participants also bring a fresh and independent perspective to the inspection process.

We are cautious about including certain stakeholders and factions in usability inspections. Unless they are particularly cooperative and open minded, marketing and sales staff can become a problem. We have found that many have difficulty stepping out of role and thinking in terms of usability as distinct from a market-driven agenda. Nevertheless, guided by a strong Lead Reviewer who can set the stage well and keep the group focused and moving forward, marketing and sales people can contribute usefully to an inspection.



Business decision makers, particularly from upper management, can also create problems for inspections. For one thing, they are apt to see an inspection that uncovers numerous defects not as a success but as a disaster in the making or a sign of incompetence on the part of the designers and developers. To the extent that they are pushing an agenda or championing a particular solution, high-level managers can distort the proceedings. In some situations, the power politics and organizational jockeying can put a damper on the process. Other participants can become hesitant to find problems when “the boss” is in the room.


On occasion, the presence of certain managers has proved to be a boon. In one case, an initial inspection of a new Web site uncovered hundreds of usability defects. So bad was this design by an outside Web design house that users were unable to complete any of the relatively simple inspection scenarios. When the findings were questioned by top management and challenged by the design house, the managers who had participated could attest to the legitimacy of the process and the correctness of the conclusions.

Users play a central role in the CUI process, and it is vital to involve at least one user or user surrogate in each CUI. However, unlike in usability testing, in which the quality of the results often hinge on the number of user subjects in the tests, one or two users is sufficient for an effective usability inspection. Although we have conducted CUIs with as many as four users, as a rule, more than two becomes awkward and offers no particular advantages.

Whenever possible, actual or potential end users should be involved, but in some situations access to real users is difficult for one reason or another. In these cases, user surrogates may be used within the inspection. User surrogates are stand-ins for real users. They may be domain or subject matter experts, former users, users of similar or related systems, trainers responsible for training users of such systems, or first-level supervisors with intimate knowledge of users. For example, in one usability inspection, when no currency trader could be freed from trading duties to participate in an inspection, a currency trading trainer served as a surrogate.

Process Roles

One of the keys to the efficiency and effectiveness of CUIs is the use of assigned and well defined roles during the inspection. The primary process



roles are Lead Reviewer, Inspection Recorder, Continuity Reviewer, and Users. In addition, other participants may have special roles and responsibilities.

Lead Reviewer.

The Lead Reviewer is responsible for convening the inspection and coordinating the preparation. The Lead Reviewer conducts the actual meeting, ensuring that the agenda and process are followed. However, the Lead Reviewer is not a facilitator in the usual sense of the term, since the goal is not to foster openness or discussion but to find usability defects. If anything, keeping the process moving forward often requires curbing discussion. Nevertheless, the Lead Reviewer is expected to draw on and involve everyone, since there are no observers in a strict Collaborative Usability Inspection. Everyone is expected to contribute to the critique. (“You haven’t said anything, Allen. What would you add?”)

Unlike meeting facilitators in other formats, Lead Reviewers in CUIs are expected to participate as reviewers themselves, particularly since they are often among the most experienced with a proven record in inspections. While it is important that Lead Reviewers contribute comments, it is also important that they participate without dominating, otherwise they risk stifling the participation of others. The Lead Reviewer has a special responsibility to ensure the full participation of users, particularly by protecting them and regulating how and when designers and developers participate.


Inspection Recorder.

The Inspection Recorder is responsible for generation of the primary deliverables of the inspection, collectively known as the Inspection Record. These include:

- Defect Log containing all identified usability defects.
- Design Addendum with suggestions about effective features to be preserved and potential solutions to identified problems.
- Process Addendum noting objections, minority opinions, and other ancillary remarks of note.
- Continuity Log identifying inconsistencies or departures from standards.

With the input and guidance of the Lead Reviewer as well as the rest of the participants, the Recorder has the responsibility for recording all

defects discovered over the course of an inspection. Identified usability defects are entered into a defect log, which may be just a piece of paper or text file, but more often takes the form of a special template. An example is shown in Figure 1. The purpose of such a template is to simplify and standardize defect recording. Specialized templates have been created and revised over the years for application software, Web applications and Web sites, and embedded systems applications that involve hardware as well as software interface design.


Constantine & Lockwood, Ltd.

SOFTWARE USABILITY DEFECT LOG

Page DEFECT ID Location ¹	Product Interface Feature ²	Recorder	Date	Description/Notes	Severity ³		
-0	button color command dialogue field function OTHER:	graphic icon item palette keystroke label menu window	message page palette screen tool window	awkward complex cluttered confusing distracting error handling OTHER:	hidden feature hidden behavior inconsistent missing nonstandard uninformative	behavior feedback layout tolerance visibility workflow	4-critical 3-major 2-minor 1-nominal ?evaluate
-1	button color command dialogue field function OTHER:	graphic icon item palette keystroke label menu window	message page palette screen tool window	awkward complex cluttered confusing distracting error handling OTHER:	hidden feature hidden behavior inconsistent missing nonstandard uninformative	behavior feedback layout tolerance visibility workflow	4-critical 3-major 2-minor 1-nominal ?evaluate
-2	button color command	graphic icon item	message page palette	awkward complex cluttered	hidden feature hidden behavior inconsistent	behavior feedback	4-critical 3-major 2-minor 1-nominal
-9	button color command dialogue field function OTHER:	graphic icon item palette keystroke label menu window	message page palette screen tool window	awkward complex cluttered confusing distracting error handling OTHER:	hidden feature hidden behavior inconsistent missing nonstandard uninformative	behavior feedback layout tolerance visibility workflow	4-critical 3-major 2-minor 1-nominal ?evaluate


Usability Defect: Any feature, function, or facet of the user interface or its organization that violates established principles of usability (e.g., visibility, feedback, etc.) or that is likely to lead to user error, delay, confusion, or the failure to complete a task.

¹ Note location of defect by screen shot or document page; identify defect by log page plus ID number, e.g., 0-8, 2-3.
² Circle or check all descriptors that apply to the identified usability defect; write in if "other."
³ Circle or check initial estimate of severity/significance of defect without respect to cost or difficulty of correction.

Form CUI-Defect S/W5 © 1995, 1997, 2000, Constantine & Lockwood

Figure 1 - Example of a template for recording software usability defects.

The Recorder and the Lead Reviewer must work as a closely coordinated duo. The Lead Reviewer will typically recap for the recorder each defect as it identified, and will frequently check in with the Recorder to make sure nothing is missed. ("Our user is hesitating, unsure what action to take. That's a defect; did you get it down?") Conversely, the



Recorder may have to ask the Lead Reviewer to slow down or wait when there is difficulty keeping up with the task of logging all of the identified defects.

Part of the record for each defect is an initial estimate of its severity. We found that it is important to capture such an estimate immediately and in context, but the inspection process soon bogs down if the severity of each defect is debated. Therefore, we assign responsibility for making the initial determination of defect severity to the Recorder. If unsure about how severe a problem is, the Recorder might check in with the Lead Reviewer or any participating usability specialist. (“Just how big a problem do you think that is?”) The Lead Reviewer may also direct the recorder regarding severity. (“So, the scenario grinds to a screeching halt. Note that whole screen as a level four defect.”)


The Recorder also keeps a separate Design Addendum for carrying forward possible resolutions to problems when participants have particular design suggestions. The Recorder also notes in the Design Addendum anything that is clearly identified in the course of the inspection as an effective feature or approach that should be preserved as the design is revised. Although the primary purpose of a CUI is to find problems not to design solutions, as these often arise spontaneously and may have value, they should not be lost but rather should be noted and fed forward to the designers for possible use.

A Process Addendum serves to hold the miscellaneous other potentially useful or important comments from reviewers. This includes objections and minority viewpoints, as well as any other comments that the Lead Reviewer or some other participant requests to be entered into the inspection record.

Another part of the Inspection Record, the Continuity Log, is usually the responsibility of another role, that of Continuity Reviewer, but in some cases it may be assigned instead to the Inspection Recorder.

Continuity Reviewer.

Maintaining consistency in appearance and behavior in a user interface design can be a daunting task. Internal inconsistencies and departures from standards have a way of creeping in at every stage. The larger the system and the more complex the design, the more likely inconsistencies are. To help ferret out inconsistencies for correction, whenever possible, we assign the role of Continuity Reviewer to some participant in the inspection. The Continuity Reviewer, like the continuity person in a film



crew, has the challenging job of keeping mental track of all the myriad details in the various parts of the user interface as they are inspected and noting any mismatches or incompatibilities. These are recorded in a Continuity Log which is generated by the Continuity Reviewer.

In addition to internal inconsistencies between different parts of the design, the Continuity Log records other kinds of inconsistencies, such as departures from industry or platform standards, as well as nonconformance with special criteria, such as Universal Design Principles, accessibility rules, or government regulations. For example, a Web site for giving access to patient information might have to adhere to HIPAA regulations as well as Web accessibility requirements. Clearly, to successfully track adherence to standards and regulations, inspection participants, particularly the Continuity Recorder, must be familiar with them. Where such criteria are crucial for success of a system, participants should be encouraged to review the appropriate documentation prior to the inspection.

The best Continuity Reviewers tend to have a certain compulsiveness about attention to detail. We often tell clients to look for the sort of person who, when they walk into a room, not only immediately note that a picture on the wall is crooked but who also feels compelled to straighten it. Although identifying inconsistencies is the primary job of the Continuity Reviewer, all inspection participants are encouraged to try and spot them, too, and draw them to the attention of the Continuity Reviewer.

Usability Specialists.

In addition to the professionals involved in the actual design of the particular user interface being inspected, it can be useful to involve independent usability professionals if available. Such persons may have any of a variety of specialties or areas of expertise relevant to a usability such as interaction design, user experience design, industrial design, graphic design, ergonomics, or human factors. In addition to participating as reviewers and identifying defects, their primary responsibility in the inspection is to assist the Recorder in classifying or characterizing usability defects and in estimating the severity of defects. Their role is as consultants, however, not as the authority or final arbiter of what is or is not a defect or how severe it is. Of course, although their views should not rule, neither should they be ignored.

User Participants.

At least one end-user or user surrogate is needed for a CUI. If no user or acceptable surrogate is available, it may be possible to assign the role to a staff member who is skilled at role playing and who prepares for the role by learning as much as possible about actual users. In our experience, however, a role-played user is never as effective for inspection purposes as an actual user.


Users have the responsibility of enacting inspection scenarios by describing what they would do at each step. They are also invited to comment first before other participants and to report any problems or difficulties they encounter. The Lead Reviewer (and other reviewers) should encourage user participants to help find usability problems and to comment freely. Reviewers should always listen carefully and try to understand user comments.

Unlike all other inspection participants, users are not expected to adhere to any rules about what they can say or how they describe problems. Users often have strong likes and dislikes that reflect their personal preferences but are not necessarily related to usability. Many will also have suggestions about how the user interface should be designed. These should be noted and recorded without comment or discussion. Things identified as problems by users but which are not, in the opinion of the Recorder, genuine defects, can be noted as “user preference.” User suggestions become part of the Design Addendum.

While users are an important resource in the process of uncovering usability defects, they are not user interface designers and should not be regarded as ultimate authorities. The key is to keep the inspection moving without allowing it to break down into a debate with the users or a defense by the designers. (“Thank you very much. We have noted the problem with purple on the Web page and have noted your suggestion for text links. Now, what would you do next?”)

Developer and designer participants.

Designers and developers who participate in CUIs as reviewers have special responsibilities that are not necessarily easy to fulfill. They are expected to find usability problems, yet they are handicapped by personal involvement and inside knowledge. The responsibility of all participants to try and find usability problems cannot be overemphasized. This need requires that developer and designer participants develop an objective attitude that enables them to look at their own work critically. We find that



prior to the inspection it can be helpful for designers and developers to spend a day or two away from the design or system to be inspected. Working for awhile on something else can make it easier to see the user interface from a fresh perspective when the inspection starts.

Some special restrictions apply to participation by developers and designers. They are not allowed to explain, justify, rationalize, or defend any aspect of their work. There are reasons behind almost every design or development decision, but the CUI is not the place for exploring or reviewing these. Explanation on the part of the designers or developers establishes them as experts who know better than users. Explanation all too easily becomes excuse. Early on in our work with inspections we discovered that prohibiting all forms of explanation or justification made usability inspections far more productive and much less contentious.

In addition, reviewers are not allowed to debate or disagree with users. Whatever a user says is allowed to stand without argument or qualification. These seemingly draconian rules have proved necessary and effective because they keep the inspection process moving and keep users engaged in the process rather than becoming intimidated or combative.

Finally, all reviewers are exhorted not to make promises to users, even implicit promises. An innocent aside can all too easily be heard as a commitment. (“Yeah, we could make that a scroll box.”) Redesigns always involve trade-offs and weighing different needs and objectives. Avoid implicitly committing the team to a particular change or solution on the fly in the midst of an inspection. Particularly where users come from or represent the customer base, implied promises can become a problem when the final system doesn’t always match up to what was discussed in an inspection. For this reason, active consideration of design alternatives during the inspection session is discouraged.

Sharing inspection roles.

When there are a limited number of participants in the inspection session, it may become necessary to share roles and responsibilities. It is common to have only one Recorder responsible for the entire Inspection Record, including the Continuity Log. In such cases, the entire inspection team carries the responsibility for continuity review. Another frequent doubling up with very small groups is to have the Lead Reviewer also serve as Inspection Recorder, since these roles are closely coupled in any case. The Lead Reviewer can also sometimes serve as Driver or Simulator. However,

carrying double duties will definitely slow down the process and often limits the attention of the dual-duty participant in terms of finding defects.

Meeting Management


In addition to ensuring that all the upfront preparation for the inspection is completed satisfactorily, it is important to get the meeting get started on time. We recommend strongly against delaying or canceling inspections for any but the most serious of circumstances because it will be all the harder to assemble the group resources the next time around.

How you start an inspection sets the tone for the whole session. The preliminaries should set the stage for a productive review by efficiently covering all the important administrative details. They should also help participants to relax and get into the right frame of mind. Issues to be addressed include:

- Welcoming participants, particularly users.
- Confidentiality and other business issues.
- Explaining the process.
- Assigning or identifying inspection roles.
- Reviewing the inspection agenda and schedule.
- Reviewing the inspection rules.
- Team building.

After everyone is welcomed, with a particular show of appreciation for the users or user surrogates, we cover any business issues, such as confidentiality or non-disclosure. Even if all participants have already signed NDAs (and this should be the case), it is a good idea to remind the group about the sensitivity of information they may acquire, including regarding the process as you use it.

We always begin an inspection with a brief review of what the usability inspection process is all about, explaining that the purpose is to help create a better end product by finding usability problems as early as possible and that we can expect to find a lot of defects. We outline the process briefly and highlight each of the roles, identifying just who is playing each role in the current inspection. Any special agenda or criteria, such as checking conformance to rules and regulations, should be mentioned, and the schedule should be highlighted. A CUI is a time-boxed process, typically scheduled for about two hours, but occasionally for as much as three.



The rules of the CUI process are also reviewed at the outset of each session. These are:

- Everyone participates.
- Users always comment first.
- Users enact the scenarios.
- Developers enact the software.
- Everyone finds defects and inconsistencies.
- Reviewers must always explain or justify defects.
- Never explain or defend designs.
- Never prompt or give hints to users.
- Never argue with users.
- Make no decisions or promises.
- Do not design solutions on the fly.
- Note all suggestions and comments and move on.
- Follow the process and stay within assigned roles.

The rules can be posted on a wall or given to everyone as a handout. We remind everyone of the definition of a usability defect in terms of what, where, how, why, and how bad, but also make clear that user participants should feel free to say anything and express it any way they want. We may also review the guiding principles of good design, such as the Principles of Good Form, or give a few brief examples of the kinds of problems we are looking for in the inspection. In our experience, the Lead Reviewer is in the best position for setting the tone of the inspection, whether as an aggressive search for defects or as a weak review. The kinds of things the Lead Reviewer notes in the course of the inspection set an example for the group.

The most important job for the Lead Reviewer is to make sure the group follows the process by enforcing the rules and maintaining the structure. The process and structure are the keys to an efficient and productive inspection. In particular, the Inspection Record should be regarded as an essential tool for keeping the process moving. Any time anyone starts a debate, suggests a design alternative, or takes exception, the Lead Reviewer should direct that it be entered into the record and then move on. (“Your point has been duly noted. Now, does anyone see any other problems here?”)

Team building.

We think of the assembled group as an inspection team and remind all the participants that we have a shared goal in finding as many defects as possible. Over the years we have used a variety of specially designed team-building exercises aimed at getting participants into the proper spirit of good-natured criticism. The purpose of these team-building exercises is to release tension while legitimizing criticism and negative comments. For example, we have one we call the “Human Factors Haka,” which takes its inspiration from the Haka, a Maori chant used by the New Zealand All Blacks at the start of their rugby matches. We get everyone to assume the proper “fighting pose,” which is not unlike the stance taken by a baseball catcher. Then, with choreographed hand movements, we have everyone aggressively chant in unison, “Ugly, awkward, inefficient interface! Ugly, awkward, inefficient interface!” An alternative is to lead a cheerleading session in which the group is led in critical cheers, such as, “Cluttered!” “Confusing!” “Unusable!” or “Hopeless!” In any case, the laughter all around starts the inspection off on the right note.

Another exercise that we always do in every CUI is related to curtailing defensive explanations from designers or developers. We remind the group that any explanation or justification of the design or implementation just interferes with the purpose of the inspection and slows the whole process down, so participants are told to imagine themselves as truckers driving an 18-wheeler with a delivery schedule to meet and caught behind a slow driver. Whenever they hear anyone in the group start to explain the rationale behind a decision or otherwise offer reasons or excuses for the design, they should reach up and pull an imaginary cord for an imaginary air horn. We have the group practice several times making a loud and suitably rude imitation of a truck air horn as they pull an imaginary cord. As with the pre-inspection team-building chants, the very silliness of the air horn action when done in the midst of an inspection defuses tensions and serves as a friendly reminder not to get caught up in defensive explanations and arguments.

Common problems and sources of failure.

When inspections do not run as well as hoped, we’ve found some common patterns to the process going astray, particularly with inexperienced reviewers or a group led by a hesitant Lead Reviewer. Common contributing factors to less-than-optimal inspections include:


- Getting distracted by design issues.
- Getting into debates about what is or is not a defect.
- Not justifying defects by reference to established principles.
- Failing to adhere to follow the rules and procedures.
- Using poorly conceived scenarios.
- Leaving the work to any usability experts who are participating.
- Expecting users to do all the work.
- Not being aggressive enough in ferreting out usability defects.

The last three of these problems are closely related and all too often lead to inspections with few results. It is a mistake to expect user participants to find the problems. They are there to provide user input and an independent perspective. Their performance of the inspection scenarios provides a plausible and organized framework within which to explore the user interface in search of problems, but the group should not be allowed by the Lead Reviewer to just stand back and observe as user participants “try out” the design. Similarly, if usability experts or specialists are present, some groups will tend to defer to them and wait to hear what “the experts” identify as defects.

Except for the last of a series of inspections in a review-revise cycle, if an inspection finds only a handful of defects, it should be assumed to be a sign of the failure of the process not the success of the design. Reviewers should be discouraged from congratulating themselves if they find few defects in a first inspection.

After the Inspection

The real payoff from a usability inspection depends on what happens after the session itself. The Defect Log needs to be reviewed and refined, clarifying descriptions, correcting mistakes in the entries, eliminating duplicates, and flagging entries that are determined to be without need for action, such as an individual user preference or idiosyncratic contribution from a particular participant. For most projects, it is helpful to transcribe the inspection record into electronic form; spreadsheets are effective for the Defect Log, document files for the other parts of the Inspection Record. For large projects, usability defects should be entered into a defect tracking system. Having the Defect Log in electronic form facilitates sorting by location and type, which can help identify problem hot spots and reveal patterns of repeated design errors.



Once the Inspection Record has been cleaned up, it is often useful to pull together the design team to go through and discuss how to address the results. At some point in the post-inspection discussions, the basic architecture of the user interface should be reexamined for possible revision, particularly when large numbers of critical defects have been uncovered. In some cases it may even be determined that the basic design is fundamentally flawed and thus a major reorganization will ultimately prove cheaper and more effective than trying to fix many separate but related problems piecemeal.

It is seldom practical to fix all identified usability defects at once, so it is necessary to prioritize defects. A quick, top-of-the-head estimate of programming/design difficulty helps. Here, too, we use a subjective 4 point scale: trivial, easy, hard, almost impossible. Along with the defect severity, this gives a basis for scheduling and assigning responsibility for working out solutions to the problems. Whether done through a formal defect tracking process or informally with a spread sheet, it is important to track whether and how all the identified defects were dealt with.

Inspections and Testing

Usability inspections should not be regarded as a substitute for usability testing. Numerous advantages come from combining inspections with testing, particularly when both are preceded by a solid usage-centered design process. Although inspections, particularly the collaborative technique described in this paper, can identify more usability defects with less effort, usability testing may uncover defects missed by other techniques. In our experience, the added payoff of actual usability testing comes in finding subtle but serious defects that may be missed by both expert evaluation and collaborative inspection. On the other side, CUIs tend to reveal numerous small problems that are likely to fall below the threshold of awareness and detection in test sessions but that collectively can substantially degrade product usability.

Our own approach to usability testing is to use it in a highly focused way to target specific issues or to answer particular remaining questions. For example, it is sound practice to verify the usability of any novel or non-standard design elements, particularly the use of custom controls in a user interface. Targeted usability testing is also particularly useful for comparing alternative schemes or approaches through simulation or using active prototypes.

Related Techniques

Although it was developed concurrently and independently, the Collaborative Usability Inspection technique is related to several other inspection and evaluation techniques. Among the best known and most closely related are Cognitive Walkthroughs, Pluralistic Usability Walkthroughs, and Heuristic Evaluation.

Cognitive Walkthroughs, originated by Lewis and Polson (see Wharton et al., 1994), are based on established cognitive theories of learning. Like CUIs, they are driven by predefined task scenarios, but typically only a single scenario per inspection. Success criteria are spelled out in advance for each step in the scenario. Because of the relatively narrow focus and highly specific concept of defect, Cognitive Walkthroughs tend to find relatively few problems for a given effort, and the problems uncovered tend to be of a less severe nature.

Another related technique is the Pluralistic Usability Walkthrough originated by Bias (1994). Like the CUI, this technique is a collaborative approach involving mixed groups of developers, designers, and users. It is another task-sensitive technique driven by a prepared scenario. However, unlike in a CUI, all participants act in the role of users, and the basic format is quite different from a CUI. For each step in a scenario, the participants independently decide on and record an action before discussion by the group. For this reason, the process can be somewhat tedious and does not typically identify a large number of defects for the time involved. Because a detailed storyboard of all screens needed within the scenario is supposed to be prepared in advance, exploration and inspection of the user interface is usually limited to a single thread through the scenario.

Strictly speaking, Nielsen's Heuristic Evaluation technique (Nielsen, 1994) is a form of expert evaluation rather than an inspection process. In practice, however, it is often lumped with inspection techniques (Nielsen and Mack, 1994). It is so well-known and widely practiced that is sometimes treated as a standard service in Requests for Proposals.

Like CUIs, Heuristic Inspection draws on a set of rules or principles. Indeed, the technique takes its names from the heuristics or rules of thumb that are used as criteria for evaluating usability. The standard approach draws on 10 heuristics derived from research on usability problems. In their original form these range from the highly specific ("clearly marked exits") to vague generalizations ("simple and natural

dialogue”). Although many practitioners still use the original set, a revised set of heuristics based on a factor analytic study is more consistent in level of abstraction and cast in language that better facilitates judgment by the evaluator.

Heuristic evaluation is a two-pass process. The first pass is an overview of the entire user interface—or as much of it as is available for inspection. The second pass evaluates individual screens or interaction contexts based on the heuristics. Unlike the CUI approach, however, it is not scenario-driven.

Conclusions

Collaborative Usability Inspections combine some of the best advantages and overcome many of the shortcomings of several other techniques. The CUI process draws on both users and experts in a structured, collaborative, scenario-driven format. It is easily learned and easily incorporated into almost any development process or life-cycle model. It offers quick turn-around of copious results.

To summarize, Collaborative Usability Inspections:

- are easy to learn how to conduct.
- fit into almost any development life cycle—or none at all.
- complement usability testing.
- can be very inexpensive.
- can be fast and simple.
- yield results quickly, early in process.
- can be used iteratively to gain more improvements.

However, it is important to remember that inspections need structure, roles, and systematic process to be efficient and effective. They also need to be guided by established and mutually agreed upon principles or rules of usability.

In addition to the immediate value of identifying usability problems quickly and efficiently for a current project, CUIs offer some significant long-term payoffs. Over time, relationships with clients and users tend to improve as they see real results from their participation. Many users are excited by being able to contribute to product improvement; they feel listened to and valued.

Improvement in a given design can be tracked through successive inspections and redesigns. Improvements and changes in the design and


development processes can also be monitored based on the results of successive usability inspections.

Designers and developers who participate in CUIs absorb basic usability principles and learn how to avoid common mistakes. Defect logs can be analyzed for trends or patterns, such as, frequency of defects by type, by designer, by location in the user interface, by design technique employed, and so forth. Such information supports root-cause analysis and can be invaluable for guiding process improvement efforts.

In short, the more you do Collaborative Usability Inspections, the more you stand to benefit.

References

- Bias, R. G. (1994) The pluralistic usability walkthrough: Coordinated empathies. In J. Nielsen and R. L. Mack, eds., *Usability Inspection Methods*. New York: Wiley.
- Nielsen, J. (1994) Heuristic evaluation. In J. Nielsen and R. L. Mack, eds., *Usability Inspection Methods*. New York: Wiley.
- Rubin, J. (1994) *Handbook of Usability Testing*. New York: Wiley.
- Wharton, C., Rieman, J., Lewis, C., and Polso, P. (1994) The cognitive walkthrough method: A practitioner's guide. In J. Nielsen and R. L. Mack, eds., *Usability Inspection Methods*. New York: Wiley.
- Seiden, J. (2003) Persona-based expert review: a technique for usefulness and usability evaluation. In L. Constantine, ed., *Performance by Design: Proceedings, forUSE 2003*. Rowley, MA: Ampersand Press.
- Chattrachart, J., Cave, D., and Vaduva, A. (2003) Learning and doing expert evaluation: A teaching dilemma. In L. Constantine, ed., *Performance by Design: Proceedings, forUSE 2003 2nd International Conference on Usage-Centered Design*. October 19-22, Portsmouth, NH, U.S.A. Ampersanda Press.
- Constantine, L. L. (2003) Canonical abstract prototypes for abstract visual and interaction design. In J. Jorge, N. Jardim Nunes, and J. Falcao e Cunha, Eds. *Interactive Systems: Design, Specification, and Verification*. Proceedings, 10th International Workshop, DSV-IS 2003, Funchal, Madeira Island, Portugal, 11-13 June 2003. Lecture Notes in Computer Science, Vol. 2844. Springer-Verlag.
- Constantine, L. L. (1994) "Collaborative Usability Inspections for Software." *Software Development '94 Proc*. San Francisco: Miller Freeman.
- Constantine, L. L. (1998) Rapid abstract prototyping. *Software Development*, 6 (11), November. Reprinted in S. Ambler and L. Constantine, *The Unified Process Elaboration Phase: Best Practices in Implementing the UP*. Lawrence, Kansas: CMP Books, 2000.

- 
-
- Constantine, L. L., and Lockwood, L. A. D. (1999) *Software for Use: A Practical Guide to the Models and Methods of Usage Centered Design*. Reading, MA: Addison-Wesley.
- Constantine, L. L., and Lockwood, L. A. D. (2001) Structure and style in use cases for user interfaces. In M. van Harmelan, Ed., *Object Modeling and User Interface Design*. Boston: Addison Wesley.
- Constantine, L. L., and Lockwood, L. A. D. (2002) Usage-centered engineering for Web applications. *IEEE Software*, 19 (2), March/April, pp 42-50.