

User Day Toolkit

[Print version \(PDF\)](#)

Introduction

1. Variants of a User Day

- [Organizational Form](#)
- [Focus](#)
- [Testing Method](#)

2. The Structure of a User Day

- [Overview](#)
- Preparation for a User Day
 - [Planning Meeting](#)
 - [Inviting the Participants](#)
 - [How Do We Create Tasks?](#)
 - [How to Interact with the Customer](#)
 - [What Do We Log?](#)
- Running the User Day
 - [Testing with a Paper Prototype](#)
 - [How Do we Build a Paper Prototype](#)
 - [Task Based Exploration together with the User](#)
 - [Performing a Lite/Lab Usability Test](#)
 - [The Usability Round Table](#)
- [Debriefing Phase](#)

Introduction

What is a User Day?

In addition to the functionality, simple to learn and problem-free software are key elements to satisfying users of software applications. To achieve this goal requires early and continuous user feedback during the development cycle. The User Day is a tool to accomplish this aim by introducing suitable user-centered tools for gathering user's feedback. The User Day will check an application's design and to systematically detect difficulties with its usage. Significant design problems can be identified and corrected before the user stumbles over them at his work place.

The User Day toolkit describes different types of user-integrated testing evaluation methods — cognitive walkthrough and paper prototype testing as well as discount and traditional usability testing. Each testing method, while optimized for a particular stage of the development cycle, can be used at any time during development and, in fact, all share a substantial amount of common concepts and procedures. In addition, the User Day toolkit outlines individual (sequential) and group (parallel) testing procedures.

The purpose of a User Day is to test an application's usability by checking if the application's structure and user interface matches the working practices of the user and if the user can use the application as was intended by the developer. User Days are carried out during the design (to aid design decisions), implementation and testing phases of software development. (see Figure 1)

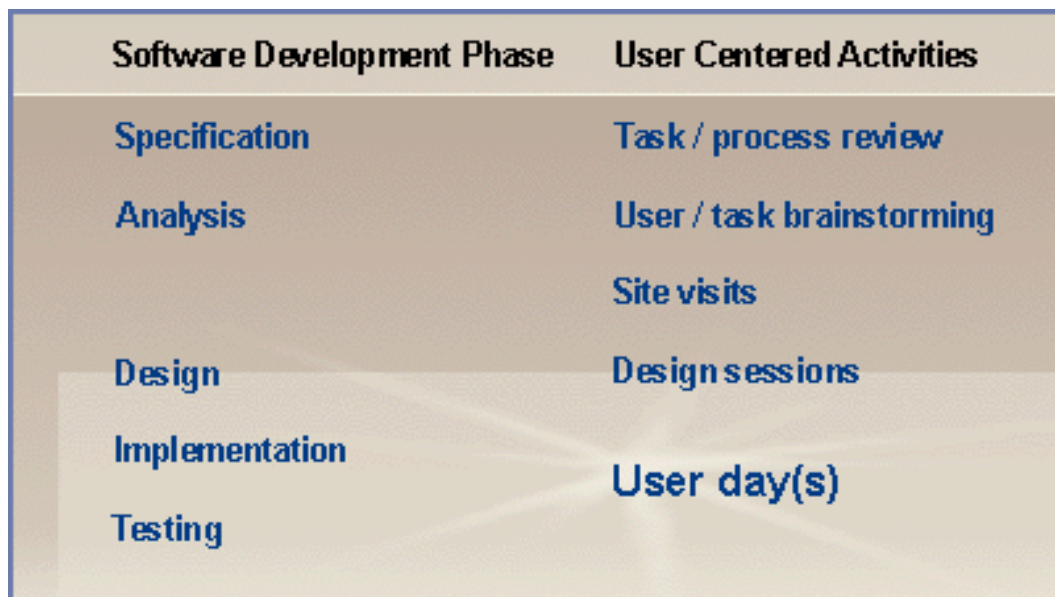


Figure 1. User Days within the software development cycle and relationship with other user centered activities.

The content and scope of a User Day depend on the status of the development project. In the early development phase, a User Day can change the basic concept of the product or the general dialog paradigm, reorganize the application structure, and optimize the rough interface design. This can help prevent the designer from clinging to a product version or design idea that does not suit the user's working practices or that the user does not understand. In the later development phase, the User Day can check the final user interface and optimize any interaction issues. In this case, the User Day resembles a classic usability test.

Finally, the intention of the User Day is to evaluate and optimize the usability of software. It is not intended to use the customer as a designer, but instead as an expert of his or her work. **A User Day is not a substitute for a good design work or strategic product planning.**

The following initial activities should therefore be carried out prior to the User Day:

Activity	Possible findings	
Usability review	Inconsistencies within the application, incorrect navigation, style guide violations, unsuitable navigation; low efficiency for standard scenarios of usage; inconsistencies in other applications within the same business process chain	4
User/task brainstorming	Explicit list of users with a description of their characteristics profile and typical scenarios	4
Site visits	Exact knowledge of the actual working practices of the users; deeper and verified knowledge of the user roles and workflow	4

Source: [User Day Toolkit](#)

Variants of User Days

The key to successful integration of user feedback into the software product design is to apply the proper methods at the right point of the development cycle. Variants of User Days differ in the testing method and organizational form. Which User Day variant to select depends on a number of factors such as: the status of the development team, the design questions to be answered, and the preferences of the team concerned.

	Internal		External
	In Lab	Outside Lab	At Customer
Single (One customer at a time)	Usability Lab Test	Usability Lite testing	Site visit with prototype or alpha testing
Parallel (A group of customers)	-	Usability Workshop	Usability Workshop

Organizational Form

Basically, there is a choice between "single" and "parallel" tests. Single tests are organized to take place consecutively on separate occasions; parallel tests are carried out together on one day. The advantage of single tests is that the developers have the opportunity to apply what they learn in the course of the tests. This organizational format is therefore particularly suitable for prototypes that are in an early stage of development. In this case, a series of interviews allows the developers to gradually optimize the prototype.

On the other hand, if a completed prototype or executable alpha version is available, then the team must decide whether to invite several users together on one day for testing (Usability Workshop), or whether to carry out consecutive single tests. The advantage of group testing is that multiple user feedback can be gathered in a very short period of time and a final Round Table helps to consolidate feedback from individual users.

The advantage of a Usability Workshop is that a team of developers must commit themselves to only a half day and the character of the workshop allows a more intensive dialog to develop with the customers about the suitability of the product. The roundtable, which follows a parallel testing phase, presents an ideal method of checking the product idea at a very early stage of development.

In comparison, single tests are easier to organize. The risk of a system failure is limited to one customer. However, the tests themselves take place over a longer period of time and the group discussion (Usability Round Table) with the customers cannot take place. The team then has to consolidate the opinions expressed in the absence of the customers.

When choosing the organizational form, the team also has to decide how close they want to emulate the normal working environment of the users or how much control they want to have over the test process. While laboratory tests offer the greatest amount of control over the tasks set and the working environment, testing outside the lab or in parallel test situations allow closer contact between the developers and users during the test phase. On-site testing has the advantage that the prototype or alpha version is exposed to externally controlled requirements of a realistic nature, arising from the customer's working context.

A decision among the various organizational formats must therefore weigh aspects such as the status of the application to be tested, the focus of the User Day, the amount of time available, the number of organizers, the proximity to the user and the proximity to the user's working context.

Variants of a User Day

Focus

The type of User Day you use also depends on the current focus of the development project. Some aspects cannot be tested at all if the user interface is not fully implemented. Therefore, the focus of the User Day depends on the stage within the development cycle. At the beginning of a project, the focus is on the overall concept and the application structure. At the end of a project, the focus shifts to the final design of the user interface and detailed aspects of the interaction.

However, it is well known that the chance of correcting major usability problems in software decreases as implementation proceeds. User days sometimes have a greater effect when used very early in the development process. For example, if the User Day is conducted during the Design phase, major problems can be eliminated before the program code is written.

Development phase	Designing	Coding	Alpha test
Typical testing medium	Paper prototype	Electronic prototype	Alpha and beta versions
Recommended testing method	Paper prototype testing Task-based exploration Usability lab testing		
Primary focus	Concept	Structure	Interface Details

Figure 2. The focus of the User Day depends on the software development phase.

Depending on the focus (conceptual, revision of the application structure, design of the GUI, details such as style guide compliance), the User Day can differ greatly and its results can only be integrated into the design if they are checked at the right time of the development cycle.

Source: [User Day Toolkit](#)

Variants of a User Day

Testing Methods

Three different testing methods are described and recommended in this toolkit: paper prototypes, task-based exploration of the application, and usability testing.

Method	Paper Prototype	Task-based Exploration	Usability Testing
Description	To test the design concept, a user interface is sketched on paper and tested.	The user explores an application as implemented together with the developer using task scenarios.	The developer observes users as they perform tasks and logs any problems.
Focus	Design concept, application structure, rough UI design.	Application structure and UI design.	Application structure and UI design including details.
Interview style	Cooperative design: Proposals by the tester are included directly in the prototype.	Working in partnership: Continual discussion about special features and problems.	Controlled support: Primarily observation, occasional interruption for assistance.
Protocol	Entire proceedings, all observations, comments and changes to the prototype.	Entire proceedings, all problems and observations.	All problems and observations, possible solution time, number of steps, errors, amount of support.
Advantages	Low cost, quick changes, supports cooperative design process.	Direct exchange of opinions with user while working on the product.	User is not subject to influence from developers. Measurable results such as processing time and number of errors.

If the design is in the initial stages of development, test with a paper prototype or with a task-based exploration. This kind of testing requires intensive communication, which encourages new design ideas and optimizes the basic concept. Here it is better to do parallel testing (i.e. testing a group of users at the same time), which includes a round table discussion at the end. If the design is close to being complete, the main focus is on detecting weaknesses and problems. Therefore, you should do individual testing so that you can observe each user more carefully. Although the focus of a User Day is closely tied to the testing medium, the team is free to emphasize the conceptual, structural, or detail level.



Development Status	Focus of User Day	Testing Method
Paper prototype 	Structure	Prototype test with focus on structure
	Interface	Prototype test with focus on visualization
Alpha version 	Concept	Exploration
	Structure	Exploration or lite testing
	Interface	Lite testing

Figure 3. The relationship between development phase and appropriate testing method.

The common theme with all testing methods is that the development team presents the user with a clear concept or design that will be confirmed, rejected or optimized by the user.

Source: [User Day Toolkit](#)

The Structure of a User Day

Overview

Regardless of the type of User Day, the overall structure remains the same. All User Days consist of a planning meeting, an application test, and a debriefing phase. Each step is described in more detail in the following chapters.

Preparation	User Day	Debriefing
<ul style="list-style-type: none">▫ Plan meeting▫ Create tasks for users▫ Select and invite participants	<ul style="list-style-type: none">▫ Introduce participants and product▫ Test the application▫ Evaluate the usability with the participants	<ul style="list-style-type: none">▫ Prioritize findings▫ Commit to implement changes

Figure 4. Steps of a User Day.

Source: [User Day Toolkit](#)

Preparation for a User Day

Planning Meeting

The usability specialist or quality manager should invite the participants to the planning meeting. The User Day checklist is used to decide when and for which areas of the application a User Day should be held.

Typical reasons for a User Day are:

- We have a new product concept and want to check how well this is received by users.
- We have an initial idea about the application and want to test its basic structure.
- We have a new interface design and first want to test it with a paper prototype before implementing it.
- We want to have our new executable system tested by several users so that we can integrate their feedback before completing development.
- We are more or less finished with the application and want to find and correct any undetected weaknesses.

Another way of getting feedback about the usability of the application from users is a customer visit with a presentation of a new prototype or a new application on a laptop.

The following aspects should be cleared up during the planning meeting:

When Should the User Day Take Place?

During planning, you should keep in mind that it could take several weeks to invite the users. Creating the tasks and preparing the test system or prototype should take only two to four days.

Whom Should We Invite?

- Use existing user roles from the user/task analysis and define the user profile.
- Define an optimal number of test users (three to six are recommended).
- Target potential customers.
- Establish who is responsible for initially contacting the customer and coordinating with the central test organization.
- Distribute the "Test Participant Invitation" check list.
- Send invitations to customers.

How Do We Test the Application?

- Decide on a focus for your User Day. Create a list of focus questions.
- Select suitable testing methods depending on the development status and focus questions (design concept, application structure, interface design).
- Establish which PC(s) or system(s) will be used for the User Day or who will create the paper prototypes.
- Establish who is responsible for the creation tasks and for inputting the required data in the system.
- Define a deadline for the creation of tasks.

Source: [User Day Toolkit](#)

Preparation for a User Day

Inviting the Participants

Use existing relationships with customers to recruit users. When contacting the customer, it is very important that you emphasize that it is not a product presentation, that the product is not finished, that it is being developed, and that only people fitting the target profile should take part.

When you invite a group of users for parallel testing, make sure that you avoid hierarchical problems within a company and competitive situations between companies.

There should be between three and six test users. This is true for single and parallel testing. In single testing, it is a common experience that the amount of new information discovered decreases as the number of users increases. If there is iterative design done between consecutive User Days, the number of testing sessions should be decided ad hoc depending on the results.

For parallel testing, the number of participants depends on the number of sessions that can be performed in parallel on the User Day. The group discussion in the final Usability Round Table will be more successful if there are more people participating. If you can only organize three parallel test sessions, you should consider having six participants test in groups of two on one machine.

If there are fewer machines available than there are participants, you can also include other activities to occupy the rest of the participants. For example, you could include an interview about working practices or present the current results of your user/task analysis.

Source: [User Day Toolkit](#)

Preparation for a User Day

How Do We Create Tasks?

Whichever type of User Day you select, you must prepare a meaningful set of tasks to be handed out to the test users in printed form. The tasks should consist of a comparable and representative set of realistic scenarios. These scenarios can be derived from existing user roles or user/task analysis.

Choose Realistic Tasks

When choosing and creating the tasks, do not let yourself be distracted by the planned or implemented functionality; i.e. you should not adjust the tasks to the application's strengths and weaknesses. The application must include realistic and common tasks so that users can detect typical problems.

The tasks should correspond to the **typical work tasks of the user** identified in the user/task analysis. The goal is to test the essential features of the application, not the rare exceptions used by only a few users or tasks that do not correspond to the actual tasks performed by the user.

The tasks should not exclude **critical or controversial parts of the application**. The tasks should cover those parts of the application that the developers themselves consider to be critical from a design or implementation standpoint. The input from the User Day can be especially helpful in such critical parts of the application.

On the other hand, it is clear that there is no point in testing tasks for which the necessary functionality has not yet been implemented. If you encounter such a "black hole" during exploratory use, you can point out to the user that this part has not yet been implemented or that the user should continue with a paper outline.

Another way of getting a realistic set of tasks is to ask the customer to bring along some of his or her normal daily tasks. Of course, this is easier with a prototype than with a development system that does not contain any customer data.

Create Well Formed Tasks

The problem should begin with a general description of a test scenario:

"Imagine you have started work at Incognito Enterprise, a newly formed Internet company. You are their web server administrator and are responsible for all web servers in the company. There are 15 servers in total, located in 3 buildings..."

The individual tasks should fit within the framework of the overall scenario.

The tasks should be formulated so that the objective of the task, but not the solution, is explained.

Correct: *You want to rename cost center 500 from "Auxiliary Personnel" to "Temporary Employees."*

Wrong: *Select cost center 500 with a mouse click and change the name from "Auxiliary Personnel" to "Temporary Employees" by choosing the option "Rename cost center" from the right mouse menu.*

Only a clear description of the task scenario and the objective, and not the instructions on where and how to use the application, should be included.

The task block should begin with a **relatively simple task** (such as displaying existing data).

It should be possible to solve the tasks **independently** (for example, even if the first task cannot be solved or is solved incorrectly, it should be possible to solve the subsequent tasks). However, you are permitted and recommended to divide more complex tasks into separate dependent parts A, B, C and so on.

The tasks should be defined **in the user's language**.

Correct: *Customer Miller has been promised a free delivery of 1000 promotional catalogs. Please enter an appropriate order in the system.*

Wrong: *Please create a sales order of type FD "Free-of-charge Delivery" and enter 1000 for the sold-to party, 1000 for the order quantity, and "Cat01" for the material.*

Keep the Tasks Short

The processing time for a task should be between 5 and 10 minutes. The total processing time for the whole session should be no more than two hours.

Create the System Data

Simply put, tasks need data, users need to interact and manipulate realistic data in order to find tasks meaningful. The data for such tasks must be maintained in the system. The tasks can only be processed correctly if the required data is in the system from the time the task descriptions are made available till the end of the User Day. Separate data records must be available for each tester, this is particularly important for group testing.

Source: [User Day Toolkit](#)

Preparation for a User Day

How to Interact with the Customer

The interviewer is faced with the difficulty of wanting to encourage the tester to voice observations and problems out loud, while influencing him or her as little as possible in finding solutions and interaction strategies.

You should encourage the tester to process the tasks cooperatively and to think out loud, but premature assistance indicating how to solve a task or pointing out errors is absolutely taboo!

The following table contains examples of remarks that can create a relaxed, cooperative atmosphere during the test without giving solutions:

Time	Cooperative Remark
After reading the task	How can we do that?
Test person is considering the task	What should we do now?
While discussing a solution	What do you think will happen if you try that?
While discussing a solution	Perhaps we should simply try it.
After an interaction	What effect did that have?
After an unsuccessful interaction	What do you want to do now?
When an error message appears	What does this message want to tell us?
Input problems (for example, tester overlooks pushbutton or makes a wrong entry in a field)	What did you think the field was for? Can we change it?
Comprehension problems (tester does not understand what is displayed)	What did you think it meant? What output did you expect?
Unsuitable workflow	Do you think you can do this directly here? How do you do it otherwise?
Missing or overlooked functions	Is that an important function? Is the name or location right? Where did you expect this function to be located?
Unknown terms	What do you call it?
General rejection	Check the reason for the rejection by testing an interpretation: "Do I understand correctly that you would prefer to have only the most important functions here and that everything else should be omitted?"

Avoid the following remarks and only give them if you cannot avoid it. In this case you should protocol your assistance.

Rule	Wrong Remark
Do not explain the interaction principle when introducing the prototype.	In our prototype, we wanted to make sure that the user can do this in such a way.
Do not give instructions.	First click on ...
Do not point out errors immediately.	Be careful, you forgot something.
Do not help with planning actions.	Before you do that, you must first ...

Do not give away entries or keys.

Just type ... in the field.

Do not cover for poor visualization.

That means ...

Source: [User Day Toolkit](#)

Preparation for a User Day

What Do We Log?

For a successful test, it is important to note down as much as possible. Record the steps the tester takes as completely as possible.

Note down all the tester's remarks and your own ideas in the margins.

During your observations, do not try to filter or compile the information, as you will lose time and information that could prove valuable at a later time.

The following table lists typical problem areas in applications. Both the test user and the observer should watch out for such problems during testing.

Typical problem areas	Examples
Task suitability	How well can you solve the task using the program? Do you get the desired result? Are there unnecessary requirements? Are flexible processing strategies supported?
Self-explanation	Can you learn the application without a long introduction? Can you figure out the functions by trying them out? Is there enough online help?
Efficiency	Are the results worth the cost? Does the application contain complicated steps? Is the application clear?
Structure	Does the organization of the application suit the task? Are there bothersome dialogs or an unnecessary context change?
Transparency	Is the application clear? Can you see where you are? Can you see what you can do?

This table tells you what things you should log when observing the user because they help in dealing with the typical problems described above:

Typical user problems	Examples
Comprehension problems	Does the user understand the task statement? Is it immediately clear what the task requires of the user? Does the tester understand the contents of the screen?
User works in an unexpected order	What steps does the user perform when solving the task? What program parts/pushbuttons are activated and in which order?
Errors and problems	Where do misunderstandings/incorrect transactions occur in the solution? Record any incorrect/inconvenient steps.
Assistance	Where did the observer have to provide assistance? What assistance did he or she give?

Exceptionally long breaks or processing times

Which program parts require intensive thought?
Which phases seem to be difficult or complicated?

Users working with the application for a long time will probably react in different ways. Note down the user's feedback about the application in addition to the objective observations above.

General observations	Examples
User satisfaction	Does the application make a good impression? Does the user feel frustrated when using the program? If yes, in which program parts? Which parts are particularly good?
User comments	How does the user react to the program and its steps? Note down any negative or positive points. Where does the program not correspond to the user's requirements? Note signals such things as sighing - non-verbal communication also indicates that the user finds the problem too complicated and demanding.
Design proposals	Does the user have ideas about improvements to the program design? Are they based on interaction with the product or ideas from other products? Is there a theme or principle that underlines the idea?

Source: [User Day Toolkit](#)

Running a User Day

Testing with a Paper Prototype

The main purpose of a paper prototype is to analyze design ideas with test users as early as possible. The goal is to present a more or less realistic design to the user, while minimizing the cost of creating it, so that the developer is still willing to reject or modify the design if necessary. Paper prototypes are an ideal test medium for iterative design.

The use of paper prototypes requires some skill and much concentration. For this reason there should always be two interviewers whose roles must be explicitly assigned before the interview:

- Interviewer who leads the conversation with the tester and handles the paper prototypes.
- Person who logs the entire interview.

The roles can be swapped after a certain time or depending on the application area, but they should not be mixed randomly.

Creating Tasks

After briefly introducing the product idea and the basic principles for using a paper prototype, you should ask the user to process concrete tasks with the prototype. Prepare written tasks derived from existing scenarios if possible.

Starting with the given scenarios, you can ask the user to execute their own variant of the scenario, as in their usual daily work. As in the contextual interview (see *Site Visits*), you can ask for concrete work processes that the user performed during the last two weeks.

Interview Technique

During the test, a partner-like discussion should evolve in which together you check how well the tasks can be processed with the prototype.

Three basic principles are important when testing with a paper prototype:

- Try to define scenarios that resemble the testers' normal working practices as closely as possible.
- Address every problem and suggest a redesign. Do not ask the tester how to do it better. If the tester accepts the suggestion and it can be integrated easily, change the prototype immediately (for example: OK, we will now draw a pushbutton here).
- Constantly try to interpret the (non-verbal) reactions of the user and to question the accuracy of your own interpretation (for example: I have the impression that you would prefer to have these two functions together in one context.)

Log

The person writing the log must record all observations, design suggestions and problems. It is also important to write down basic findings about the tester's working practices. The prototype can be used here as a guide.

Important feedback from the tester should be recorded on the prototype itself.

Source: [User Day Toolkit](#)

Running a User Day

How Do We Build a Paper Prototype

A paper prototype is a user interface sketched on paper that describes a planned application so completely that practically all the relevant dialog steps can be shown.

A square card represents the screen background. All the other dialog elements are made of pieces of paper and post-it notes and are placed on this card. Typical dialog elements are:

Element	Description
Window	Large piece of paper or post-it note.
Pull-down menu	The name of the menu belongs on the main window because it should always be visible. The menu itself is represented with a small post-it note.
Pushbutton	Permanent pushbuttons should be represented with a small post-it note, and not be drawn directly in the window. This permits easy re-design during the test.
Pop-up window	Another card that is placed over the main window if needed.
Radio button	Radio buttons are simply drawn in the window. The button currently selected can be simulated by covering it with a transparency with a dot on it.
Table	A table can be represented with vertical strips of post-it notes that are glued to a card. This allows you to change the order of the columns during the test.
Selected entry	Every type of temporary cursor selection can be visualized with transparency. You only have to draw a frame around a selected entry on the transparency.
Input field	Real user entries are simply written in input fields. Use removable tape for these fields to permit multiple entries.
Other data	It is advisable to create a version of the prototype containing sample data. You can thus discuss it with the user at any time. However, since "real" data is to be visualized, you should also prepare empty versions in which you can write data during the test.

Ideally, the team works together to create a paper prototype. If this is not the case, you must first show the prototype to the other developers and explain it to them before using it in a test.

If there are several interviews running in parallel during the User Day, make sure that the prototypes can be duplicated using a copier.

You should practice the interview with colleagues prior to the User Day to prevent important parts of the application from being omitted.

Define the objective of the test prior to the actual interview. You can also determine the test objective from the status of the prototype:

- The more detailed a prototype is, the better the user interface characteristics will be tested.
- The rougher a paper prototype is, the better the application structure or basic dialog paradigm will be tested.

Running a User Day

Task Based Exploration with the User

To explore an application together with a user, the application development must have reached a stage where substantial parts of the application are ready to run using test data.

The advantage of a joint task-based exploration over the Classic Usability Test is that you can discuss problems with the user directly in the context of the current task. This gives authentic and accurate feedback and comments about the usability of the application.

In contrast to paper prototypes, you cannot make spontaneous design changes here. You can only record suggestions, not implement them. In contrast to Classic Usability Testing, you have to dispense with a controlled test situation in favor of a more intensive and continuous exchange of opinions.

Creating Tasks

In task-based exploration, prepared tasks are used as a guide for exploring an application.

A good overall scenario is important and helpful because exploration is more natural if the individual tasks fit together.

In contrast to the Classic Usability Test, you can leave the normal task path at some points to explore other aspects of the application, but make sure that processing the tasks does not end up as free exploring. If it does, point out that free exploration will be possible at the end.

After processing the tasks, and perhaps after processing each area of the application, ask the user to try out the application again in his or her preferred way or how he or she would normally use it.

Interview Technique

When jointly exploring an application, the interviewer must take advantage of the fact that he or she is participating directly in processing the tasks and can therefore directly address problems as they occur. Under no circumstances, however, should he or she take the lead, suggest solutions or explain the application. He or she is thus a passive partner with regard to processing the tasks, but an active partner regarding problems that occur.

In addition to directly discussing the problems that occur, the interviewer should observe the tester closely and notice any non-verbal reactions.

The interviewer should immediately follow up on all suspicions about the tester's satisfaction and all interpretations about user problems. The interviewer has a unique opportunity to have his or her suspicions confirmed or rejected immediately by the user. The quality of the user feedback is always better the closer you are to a real working situation. This is also true for probing spontaneous design ideas resulting from the observations. Instead of keeping design ideas and interpretations in your mind, you should communicate them immediately to confirm them in the context of usage. The graphic below illustrates this principle of checking ones own interpretations and ideas with the user during the observation.

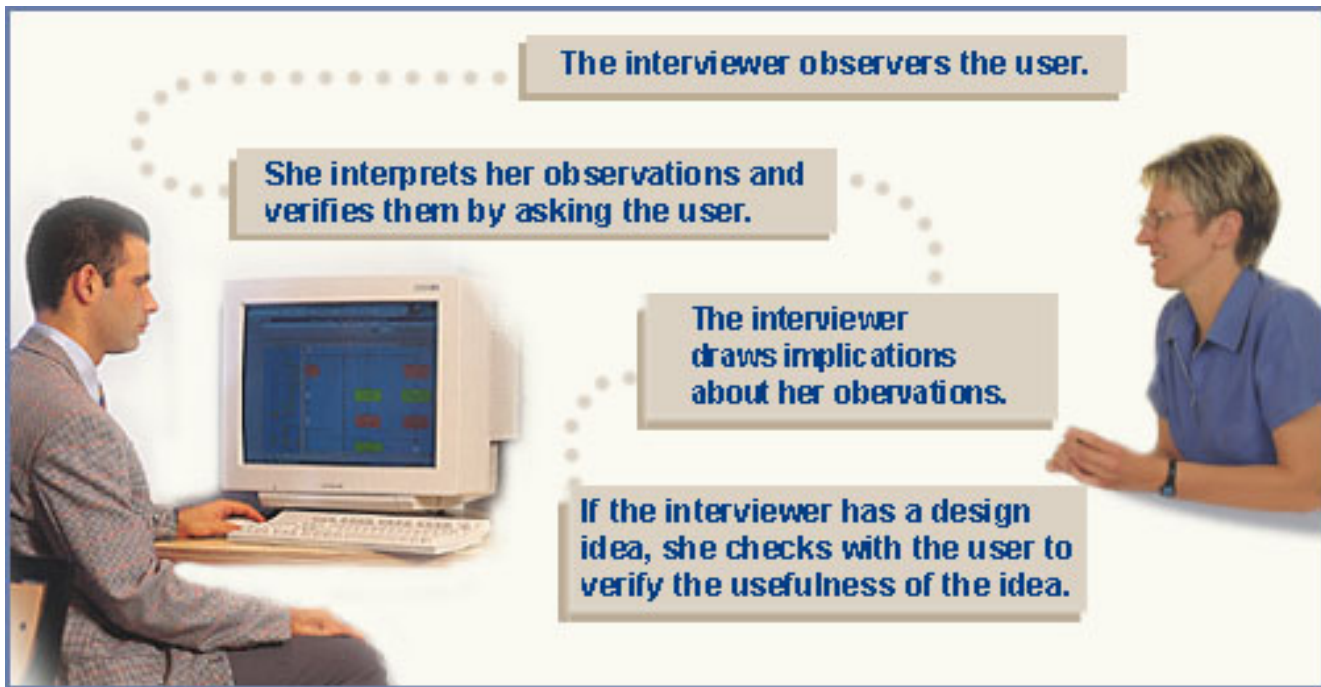


Figure 5. Check any interpretation and ideas with the user during the interview.

Log

Write down as much as possible. This is true for the testing sequences and for all important statements made during the conversation. The log should contain not only your interpretation or summary, but also the original quotes made by the tester.

Source: [User Day Toolkit](#)

Running a User Day

Performing a Lite/Lab Usability Test

In contrast to task-based exploration, the Usability Test is a controlled test situation without a continuous exchange of opinions. The user is asked to solve the tasks on his or her own, working on a problem until he or she either explicitly requests assistance or obviously cannot continue without help. The Classic Usability Test also requires an implemented application, most of which can be explored with test data, but not jointly with the observer.

The advantages of the Classic Usability Test lie in the minimal influence on the user by the observer. You can see how far the users get on his or her own without any assistance to see whether or not the application is self-explanatory. Classic Usability Testing provides quantitative results that reveal information such as the solution time, the number of work steps, the number of errors and the amount of assistance given.

The developer's primary role in this phase is to observe the user. He or she should log questions that arise as well as difficulties with the application. The developers only provide assistance during the test phase if the user explicitly requests this. The user should work alone or with a partner when testing the application.

Interview Technique

There should be only one primary contact person (the observer) for each user. The role of other attendees should be explained. More than one observer per user or pair of users could bother the testers. The observer should first introduce himself and then point out that the user isn't being tested to see if he or she makes mistakes, but that the objective of the test is to detect potential weaknesses in the application in order to improve it.

The observer in the Classic Usability Test has a purely passive role and should only interfere in task processing if the user requests help.

No direct promises for concrete changes should be made and no function requirements should be discussed during the test.

If two users are solving the problem jointly, the observer should make sure that the users swap roles after each task.

Exceptions

The observer should only actively intervene while the customer is working when:

- He or she notices that the tester is losing too much time in an unplanned part of the program.
- He or she notices that the attitude of the tester threatens to become negative because of difficulties in solving the tasks.
- He or she notices that the tester cannot solve the problem without assistance.

If the user needs assistance, the observer should note what aspect of the program led to that point. Then ask the user to comment on the problem they are experiencing (for example, "Please tell me where you are stuck" or "Which function are you looking for?"). Often this results in the user solving the problem for him or herself. Obviously, the observer should never make negative comments that could disturb the user. For example, he or she should never say, "You did the wrong thing" or "You should not have used Function XYZ."

Log

In the Classic Usability Test, the observer plays a passive role and can concentrate on logging the test situation, writing down everything he or she notices. The observer should carefully record difficulties and problems requiring his or her intervention and

to log the type and scope of his or her assistance.

Discuss the notes with the users after processing the individual tasks and correct them if necessary. This step should make the log more transparent for the user. If you find that you need to discuss a certain point, you can refer to the discussion in the afternoon.

Source: [User Day Toolkit](#)

Running a User Day

The Usability Round Table

The Usability Round Table is a separate focus-group-like discussion at the end of the parallel testing phase. Users have the opportunity to exchange their experiences and have a chance to discuss with the development team their concerns and needs. For the development team this is a great chance to establish a first consolidated list of the major issues revealed in the User Day.

Introduction

The Usability Round Table collects users' impressions of the application. It can be applied to all testing methods that are performed in parallel sessions in one User Day with multiple users.

Typically the Usability Round Table takes up the last one to two hours of the User Day and users share their experiences of the software application with the development team. The moderator of the Usability Round Table draws up a plus-minus list for the tested application. The entries on the minus side are discussed in detail and suggestions are collected as to how these points can be avoided or corrected. At the end, the users can optionally fill out questionnaires on the application tested and about the User Day method in general.

Participants and Roles

To avoid giving the users the inhibiting feeling of being watched, the discussion group should not be too large. Therefore the users and moderators should outnumber any other participants. With the recommended number of six users, one to two moderators, and about three to four members of the development team, there should not be more than ten to twelve participants. Developers should mainly be passive observers. Their task is to help the moderator if application-specific questions come up.

Users

The users report their findings about the application being tested and make contributions to the discussion.

Moderator

A moderator supervises the Usability Round Table and leads the discussion. You can also have a pair of moderators instead, in which case they also share the task of keeping the log (see *Logger*).

The moderator initiates the discussion with his or her questions and is responsible for documenting the contributions to the discussion in a comprehensible form and language so that developers who are not participating can understand them. A usability coordinator is suitable for this role since both expertise in user-integrated procedures and knowledge of the application and its context are combined here in one person. Training in moderation techniques is recommended.

Logger

This person writes down keywords from the discussion. For example, the reasons for and explanations of the user contributions are important since they are not recorded on post-it notes or cards and are only mentioned in the conversation with the participants. The log can be used to inform interested persons and members of the development group not taking part in the discussion about the contents and results. Important detailed information contained in the conversation should also be recorded for use in the design revision. For example, a user rejects the planned use of a drop-down list box for a field because too many entries appear. Until receiving this information, the developers might assume that not more than ten entries would be necessary at a company.

Developers

The developers take a passive part in the discussion. The developers themselves do not make any contributions to the discussion, but are available for consultation. If a developer does not understand the user's contribution to the discussion, he or she asks questions for his or her own understanding. He or she may not comment on the user's statements, on the functionality, or on the design of the user interface in future versions of the application you are testing.

Other Stakeholders

People outside the development team with an interest in the results of the User Day can participate in this discussion. Typically, documentation writers, quality managers or sales and marketing people might be involved. They act in the same manner as the developers.

Objectives of Usability Round Table

The discussion has the following objectives:

- The experiences of users during the testing sessions are communicated, collected and handled in a structured manner. It is important that the users say as much as possible to provide the development groups with sufficient information. The objective is to generate a large number of individual contributions without having to worry about the qualitative basis of each individual statement.
- The qualitative aspect of individual user opinions is tested automatically in the Round Table, as the reaction of other test users will indicate whether it is really an individual opinion or a group opinion.
- The users discuss aspects that were not mentioned when working with the system, which creates more useful material for the development group.
- Contradictory statements and different user opinions can stimulate discussion. Record the reason for each opinion expressed and assessment made. The objective is not to find the "right" opinion.
- Depending on the focus of your User Day, you should emphasize topics related to either the conceptual, the structural, or the detailed user interface design.

The discussion does not have the following objectives:

- The discussion is not about functionality that is missing, would be useful, or is planned for the future.
- No decisions or promises should be made about open design questions during the discussion, but you should encourage discussion of a solution from the point of view of the user.

Resources Required

- The room for the discussion should be equipped with a (mobile) projector and a computer so that any statements made by the users can be examined using the application you are testing.
- Use at least two flip charts for comments on positive and negative experiences.
- Provide pens for all users.
- Check (and ask the users while preparing for the User Day) whether a video camera may run for the purposes of logging the results. The video camera is focussed on the pin boards and is primarily for documenting the conversation. The advantage of using a video recording is that it is no longer necessary for someone to write the log.

Schedule

One to three hours are needed for the discussion; Typically this begins in the afternoon when the users and other participants return from the lunch break. It is also possible to schedule parallel test sessions and subsequent Round Table in half a day.

Introduction

In this phase, the moderator explains the general course of the discussion, the objectives, and the desired results. The

participants also introduce themselves if required.

- **Duration:** 15 minutes.
- **Material:** Name cards, pens.

Creating the Plus-Minus List

The plus-minus list is used to structure the discussion. The procedure is extremely simple for both the moderator and the users and helps to structure the Round Table. The moderator lists all the positive and negative aspects expressed by the users on the flipcharts. You should allow 15-20 minutes for this.

The moderator should first collect all positive aspects experienced in the test sessions. Although you may switch between positive and negative aspects, it is usually hard to return on positive aspects once you start to discuss negative aspects.

If collection of positive aspects slows down you may move on by collecting negative aspects. The negative aspects are the starting point for a discussion of the solution in the next phase. Consequently, you should clearly separate the collection of negative points and the discussion about possible solutions.

The moderator points out to the users that ugly and uncomfortable aspects are also negative aspects that would otherwise be considered to be unimportant. The objective is to gather as many aspects as possible in the plus-minus list. Quantity has priority over quality or (apparent) irrelevance of a user experience.

By asking appropriate starter questions the moderator has the control and responsibility to influence the collection of plus/minus points into a direction which fits to the focus of the User Day. The moderator can stimulate the feedback by asking prepared questions that focus on concept, structure, functionality, and user interface detail of the product. You may also use the terms of ISO 90421 Part 10 to structure the discussion. It is a good idea to hand out those questions as general focus questions prior to the test sessions to encourage the test user to focus on such problems.

To support the authenticity of individual opinion against group dynamics, you may also ask users to write down their personal plus/minus list first on a paper before collecting those points all together.

Finally, you should take care to check and log how many users agree with this point.

Discuss Negative Aspects and Brainstorm Possible Solutions

If there are a lot of negative aspects, you should prioritize them, i.e. each user should mark the two aspects that he or she considers most important. You can then deal with the aspects according to the number of times they are marked.

The discussion should result in a consolidated and comprehensive problem statement for each negative point so that interested persons who are not participating in the User Day can read about the discussion and use the results. Adhere to the general rules for collecting feedback, i.e. the users' remarks should not be commented upon and developers should only ask comprehension questions.

- The moderator discusses each negative aspect in turn and asks all users to explain if and why this is a negative point. The person writing the log (second moderator) notes down this explanation.
- The user who submitted this negative point might be asked to explain the problem again to create a shared understanding. In no way should there be any evaluation about the quality of the feedback. Questions from the developers to increase their comprehension are particularly welcome.
- After having a clear understanding about all negative aspects, the discussion of solutions begins. The moderator must keep an eye on the time schedule and use the time available efficiently (normally there is approximately 60 minutes). The objective is to discuss proposed solutions for every problem identified. It is very important that the moderator divide the discussion based on the different types of problems (concept, structure, interaction design, visual design) and that he or she avoid long discussions about things that cannot be decided anyway. Try to stay close to the original problem as occurred in the test session.

- If the users have no proposals for a solution at the beginning of the discussion, the moderator can prompt the user by asking the following questions:
 - How they would have expected the application to work?
 - What would have helped them handle the problem?
 - How the application could have helped more?
- The users discuss these solutions in the presence of the developers. Not only the results of the discussion but also the different contributions and ideas for solutions are recorded. Other users are asked to present a proposal for a solution. The developers also provide spontaneous ideas for a solution and ask whether the solution would solve the user's problem.

At the end of the discussion, the users or the moderator should note the proposed solution or alternatives and competing solutions should be noted briefly.

- The moderator makes sure that the developers do not discuss whether the suggestions for improvement can really be implemented. The primary aim is to collect as many solutions as possible from the user. Future (planned) release versions should not be mentioned either. In short, the discussion should not go into details.
 - **Duration:** 90 minutes
 - **Material:** Pens

Summarizing and Saying Goodbye

The summary gives the users a final opportunity to comment of the application.

- The moderator gives a final overview about results of the discussion. The plus list is used to mention that there were some positive aspects as well. The negative list is used to outline the opportunities of the product.
- The moderator thanks the users for their participation, gives them a present, and says goodbye. He or she may also explain the further activities, if any are planned in the end phase.
 - **Duration:** 15 minutes
 - **Material:** Cards, for each user, pens, pins, thank-you gifts for participants

Additional Phase (Optional)

After the User Day is finished, many development teams use the time to enhance the user's experience of the application. For example, the development group can present planned new developments or this phase can also be used to ask the users what they thought of the User Day. Further marketing and sales aspects can also be included here, for example special presentations, a product-related questionnaire or an overall evaluation of the user interface.

Source: [User Day Toolkit](#)

Debriefing Phase

The purpose of the Debriefing phase is to process the results of the User Day and incorporate findings in the development of the product.

What to Bring to the Debriefing Phase

- Tests logs.
- Further notes containing observations, problems, ideas and open questions.
- Log from the Usability Round Table with user feedback.
- Plus/minus list from the Usability Round Table with suggested solutions for the negative aspects.

Discussing the Test Results

The presentation of the course and results of the parallel tests should be sorted so that you can see how many users had a certain problem in each area of the tested application.

Sort the problems by their importance (central function, typical task scenario) and frequency, collect the solutions, and discuss them in order of importance.

Usage problems can occur at the task level (goal level) or operating level (system interaction). Both types of problem can occur as long as you are testing with a set of realistic tasks. It is important to recognize the level at which a problem occurs and its origin, so that you know what you are discussing (concepts, structure, interaction design, or visual design):

Problem description	Problem level
When the user returns from a dialog window, the last data objects selected are no longer selected. The user thus loses his or her current work context and is forced to select the same object again and again.	Detailed Interaction Design on application level
While working, the user expresses a desire to have an overview of the results of his or her work.	Structural Interaction Design Mismatch with information needs
A user does not use a function even if it was on the screen.	Visual/Interaction Design
A user explains that this application would not be of value for his or her work unless it is able to display aggregated data.	Conceptual Design
A user complains about the 3D-look of the buttons.	Visual Design
The user has problems scrolling in the table control.	Detailed Interaction Design on generic control level

Consolidating Other Notes

During the User Day a lot of notes are collected, not all of which can be discussed individually. Some of them will overlap with observations made by other colleagues.

You should therefore create clusters of related observations together with the team. The recommended method is to copy the observations on post-it notes and create an affinity diagram. This is done as follows:

- All participants stick their post-it notes on a board in no particular order.
- All participants read the post-it notes, remove post-it notes with similar remarks, and hang them in columns on another board.

- This is done until all the post-it notes have been transferred from one board (unsorted) to the other. Of course, existing columns must be taken into consideration.
- The columns are then assigned titles, and unsuitable remarks are removed from the columns. A good column size is 3-6 post-it notes.
- You should combine several columns to form one term of a higher level.

Only the clusters or problem areas are discussed at this point. Direct design ideas will always be developed during the tests and discussion. You can now analyze them again for consolidated problems.

The rule for ideas and solutions is: the higher a solution is placed in the affinity diagram, the better it is.

Updating the User/Task Analysis

If the test and discussions produced large deviations to the user roles and scenarios existing in a user/task analysis, these should be enhanced or changed. This should be done with care because direct observations from customer visits are always more realistic than what the customers report about their work.

Source: User Day Toolkit