UX Research and Design: From User Needs to Usability Testing

Anna Onyskiv

Department of Software, Lviv Polytechnic National University, UKRAINE, Lviv, 12 Stepan Bandera Street, E-mail: anna.onyskiv.pz.2025@lpnu.ua

Abstract. Usability problems often arise when developers make interface decisions without knowing how people will use the product. This paper explores how user research, personas, user flows and prototype testing can help identify these problems. A university room-booking application is used to illustrate the process.

Keywords: UX research, persona, user flow, prototyping, usability principles, user-centred design, Fitts's law, Hick's law.

I. INTRODUCTION

Imagine a student who needs to book a room for a group project. The university has an online booking service, so there should be no need to contact the administration or visit the building in person. However, finding a suitable room is not always straightforward. A student may select a room that appears to be available and discover that the reservation cannot be completed. In another case, the necessary filters may be difficult to find, or the booking process may contain more steps than expected.

There may be no technical failure in either situation. The problem is that the service does not make the task easy for the person using it. This is one of the reasons why software quality cannot be judged only by the number of available functions or by whether the system works correctly.

User experience (UX) design considers how people interact with a product and whether it meets their needs in a particular situation. ISO 9241-210 [1] describes a human-centred approach in which users and their requirements remain part of the development process. Hassenzahl and Tractinsky [2] take a broader view of user experience, pointing out that it involves not only the practical use of a product but also people's expectations, feelings and the context of interaction.

In practice, developers often have to make decisions before they know exactly how users will behave. A feature that seems obvious to someone familiar with the system may be confusing to a first-time user. This makes it necessary to collect information about users before deciding what the interface should look like.

The research question of this paper is: How can user research, personas, user flows and prototype testing help reduce usability problems when developing a university room-booking application?

The paper argues that these methods are more useful when the results of one stage influence the next. Research provides information about users, personas and user flows help turn that information into design requirements, and prototype testing allows the team to check whether its decisions work. The following sections examine this process and its application to a room-booking service.

II. USER RESEARCH: FINDING OUT WHAT USERS NEED

A development team may have a clear idea of what its product should do, but this does not necessarily mean that it understands the people who will use it. Before designing an interface, the team needs to find out what users expect, what difficulties they experience and what they are trying to achieve.

There are several ways to collect this information. Interviews, observation, contextual inquiry and surveys are among the methods commonly used in UX research. They do not provide exactly the same results, so the choice depends on what the team wants to understand.

Interviews are useful when researchers need to hear about people's experiences in their own words. For example, students could be asked how they usually find a room for group work, how far in advance they make reservations and what information they look for before choosing a room. Their answers may reveal that room capacity, location or available equipment are more important than the development team initially expected.

However, what people say does not always match what they actually do. A student might claim that booking a room is easy but still struggle to find the correct option when using the service. This is where observation becomes useful.

Beyer and Holtzblatt [3] describe contextual inquiry as a method in which researchers observe people in their usual environment and discuss their actions with them. In the case of a university booking service, a researcher could watch a student trying to reserve a room. The student might repeatedly return to the timetable to check available hours or open several pages to compare rooms. These actions can reveal problems that would probably not appear in a short interview.

Surveys provide another perspective. They can be sent to a larger group of students and used to find out how often certain problems occur. For instance, a survey might ask whether students have difficulty finding available rooms or whether they would prefer to see equipment information directly in the search results. The answers can help establish whether an issue reported by one participant is common among other users.

The value of combining these methods becomes clearer when their limitations are considered. An interview gives detailed personal information, but it may not represent the experience of everyone. Observation shows actual behaviour, although it usually involves fewer participants. A survey reaches more people but may not explain why a particular problem occurs.

Nielsen [4] discusses the use of small-scale usability studies during the design process. His work draws attention to the fact that teams do not always need a large study before making improvements. Smaller tests can help reveal important problems while the product is still being developed.

Once the information has been collected, it needs to be organised. One possible technique is affinity mapping. Researchers write down individual observations and comments, then group similar ones together. In the booking application example, several comments might relate to searching for rooms, while others concern unclear availability information or difficulties cancelling reservations.

At this point, the team should look for repeated problems rather than focus on every individual comment separately. If students regularly have trouble checking room availability, this issue may deserve more attention than a suggestion concerning a minor visual detail. The findings can then be used to describe the main user groups and the tasks the application needs to support.

III. PERSONAS AND USER FLOWS

After research, the team may have a considerable amount of information but still need a way to use it during design. Personas provide a more understandable picture of the people for whom the product is being developed.

A persona is a fictional user profile based on research findings. It usually describes a person's goals, behaviour, needs and difficulties. Cooper [5] introduced a goal-oriented approach to interface design, emphasising the importance of understanding what users want to achieve. Pruitt and Grudin [6] later examined how personas can help development teams communicate and apply user research.

For a university room-booking application, the research might suggest two common situations.

- **A student who plans ahead** usually books a room several days before a group meeting. This person wants to compare different rooms and check their capacity and equipment.

- **A student who needs a room immediately** searches shortly before a meeting. For this person, finding an available room quickly is more important than comparing every possible option.

These profiles are simplified examples, not descriptions of real participants. Their purpose is to show that users may approach the same service with different priorities. A design that supports detailed comparison should not make quick bookings unnecessarily difficult.

It is also important not to invent persona details without evidence. Adding an imaginary age, personality or personal background does not automatically make a persona useful. The information should relate to the way a person uses the product and should come from research.

While personas describe users, user flows describe the actions they take to reach a goal. Garrett [7] discusses the relationship between user needs and the structure of an interface. A user flow makes this relationship more concrete by showing the sequence of screens and decisions involved in completing a task.

For example, a student booking a room might follow this sequence:

1. Open the application.
2. Select the required date and time.
3. View the available rooms.
4. Choose a room that meets the requirements.
5. Check the booking details.
6. Confirm the reservation.

This sequence describes the expected situation, but it does not cover everything that can happen. Another student might reserve the selected room before the first student confirms the booking. The application then needs to explain why the reservation is no longer possible and allow the user to return to the list of available rooms.

Such situations should be considered during flow design. Otherwise, the team may create an interface that works only when everything goes according to plan. Error messages, alternative choices and ways to return to the previous step are also part of the user's experience.

Table 1. How UX artefacts are used during design

Artefact	**Question it helps answer**	**Result**
Research findings	What difficulties do users experience?	A list of needs and recurring problems
Persona	Whose needs should the design support?	A research-based user profile
User flow	What steps does the user take?	A sequence of actions and possible alternatives
Prototype	Can users complete the task with this design?	Feedback for further improvements

Table 1 shows how each artefact contributes to a different part of the work. However, their content may change as the project develops. For example, if testing shows that students regularly skip a particular step, the team may need to reconsider the original user flow. The persona may also need revision if the research was based on an incomplete understanding of users.

IV. PROTOTYPING AND TESTING DESIGN IDEAS

Once the team has a clearer picture of its users and their tasks, it can start exploring possible interface solutions. Building the complete application immediately would take time and could lead to unnecessary rework if the initial design turned out to be unsuitable. A prototype offers a simpler way to examine an idea first.

A prototype is an early representation of a product. It may be a paper drawing, a set of connected screens or an interactive model. Houde and Hill [8] explain that prototypes can be used to examine different aspects of a system, such as its role, implementation or interaction. This means that a prototype does not always need to look like the finished product. It only needs to help answer the question the team is currently considering.

A paper prototype is often enough when the team wants to compare different layouts. For example, designers could sketch two versions of a room-search page and ask students where they would expect to find filters for capacity or equipment. Rettig [9] discusses paper prototyping as a practical way to test interface ideas without first writing the application code.

When the general layout has been decided, the team can create a clickable prototype. Participants can move between screens, select options and follow the booking process. Although such a prototype may not store real reservations, it can still show whether the order of actions makes sense. The relationship between the stages discussed so far is shown in Fig. 1.

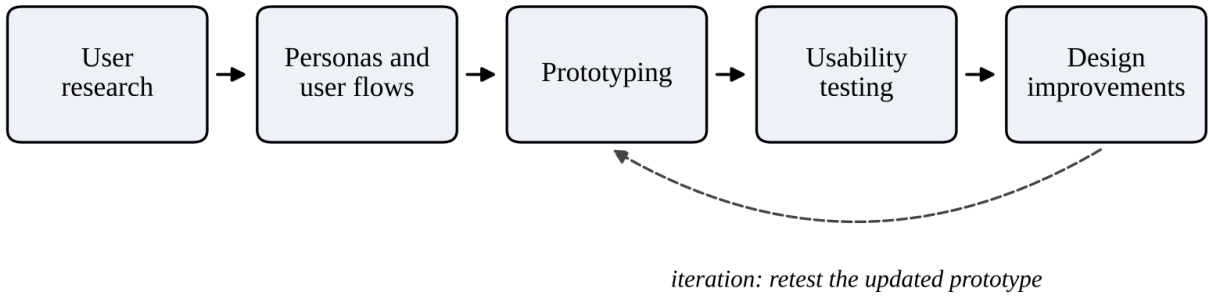

Fig. 1. UX workflow: from user research to design improvements

Testing should involve tasks that resemble situations users might encounter in real life. Instead of asking participants whether they like the interface, the researcher could ask them to book a room for a group meeting on a particular date. This gives the team a chance to see whether the participant can find the necessary information and complete the reservation.

One technique that can be used during testing is the think-aloud method. Participants are asked to describe what they are thinking while using the prototype. They might explain why they choose a particular room, what they expect a button to do or why they are unsure about the next step.

The researcher should avoid explaining the interface or suggesting where to click. Otherwise, the participant may complete the task only because they received help. It is more useful to observe where they hesitate, which elements they overlook and whether they make mistakes.

For example, a student might reach the final booking screen but fail to notice the confirmation button. Another participant might select a room without realising that the displayed time is already occupied. These observations point to specific interface problems that can be corrected.

After the test, the team should decide which changes are necessary. Some problems may require a different page layout, while others could be solved by changing a label or adding a message. The updated prototype can then be tested again.

A test does not always confirm that the original design was correct. Sometimes it shows that the team misunderstood the task or made an incorrect assumption about user behaviour. This is still a useful result: it gives the developers a reason to reconsider their decisions before the application is implemented.

V. USABILITY PRINCIPLES AND INTERFACE DECISIONS

Observing users helps designers understand where difficulties occur. However, it is also useful to have principles that explain why certain interface elements may be easier to use than others.

ISO 9241-11 [10] describes usability in relation to specified users, goals and contexts of use. In other words, a product should be evaluated according to whether its intended users can achieve their goals in the conditions in which they use it. An interface that works well for an experienced administrator may still be confusing for a student who opens it for the first time.

Nielsen's usability heuristics [11] offer practical guidelines for interface evaluation. They include visibility of system status, consistency, error prevention and recognition rather than recall. Norman [12] also examines how people understand and interact with designed objects. His work is relevant to software because interface elements should make their possible actions understandable.

Consider the confirmation stage of the booking process. After a student selects a room, the application should display the date, time and room details before the reservation is completed. Once the student confirms the action, the system should clearly indicate whether the booking was successful. Without this feedback, the user may not know whether they need to repeat the action.

Two additional principles help explain how the size, position and number of interface elements can affect interaction.

Fitts's law describes the relationship between movement time and the distance and size of a target. One commonly used form of the equation is given in Eq. (1):

$$MT = a + b\,log_2\left(\frac{D}{W} + 1\right) \tag{1}$$

where:

- MT is the movement time;
- D is the distance to the target;
- W is the width of the target in the direction of movement;
- a and b are constants determined by the task and conditions.

Eq. (1) suggests that a larger target is generally easier to reach than a smaller one, especially when it is close to the user's current position. In a room-booking application, this matters when placing the main confirmation button. If the button is too small or difficult to reach, users may overlook it or select the wrong element.

Fitts [13] originally examined the relationship between movement time and target characteristics. MacKenzie [14] later discussed the use of Fitts's law in human-computer interaction, showing its relevance to the design of interactive systems.

Hick's law considers the effect of the number of choices on decision time. A commonly used form is given in Eq. (2):

$$RT = a + b\,log_2(n + 1) \tag{2}$$

where:

- RT is the reaction time;
- n is the number of equally probable alternatives;
- a and b are constants.

As the number of choices increases, users may need more time to decide. Hick's research [15] provides a basis for considering how the presentation of alternatives affects decision-making.

In the booking application, a long list of rooms may be difficult to navigate, especially when users are looking for a particular capacity or piece of equipment. Filters can reduce the number of

options that need to be considered. However, removing too many choices is not necessarily helpful. Users should still be able to see alternatives that may suit their needs.

Neither Fitts's law nor Hick's law, given in Eqs. (1) and (2), can determine the best interface on its own. They describe particular aspects of human interaction and should be considered alongside user research and testing. Their main value is that they give designers a reason to examine details that might otherwise seem unimportant.

VI. APPLYING THE METHODS TO A ROOM-BOOKING APPLICATION

The previous sections discussed research, personas, user flows, prototyping and usability principles separately. Their practical value becomes clearer when they are applied to the same project.

Consider a hypothetical team of three students developing a mobile application for booking university study rooms. At the beginning of the project, the team assumes that students mainly need a simple list of available rooms. However, this assumption has not yet been checked.

The team could start by interviewing students who have used the existing booking service. It could ask how they search for rooms, what information they need before making a reservation and what usually causes difficulties. Observing a few students using the current service would provide additional information. For example, some might switch between several pages to compare room capacity, equipment and available time slots.

After collecting the findings, the team could group similar comments. If several students mention that they cannot easily compare rooms, this may indicate that the search page needs to display more information. If others report that they are unsure whether a reservation has been completed, the confirmation stage deserves attention as well.

The team could then create two personas based on the main patterns in the research: a student who plans bookings in advance and a student who needs a room at short notice. These profiles would help the team consider both situations rather than design the application around only one type of user.

Next, the team could prepare a user flow. The main sequence would include selecting a date and time, viewing available rooms, choosing an option, reviewing the details and confirming the reservation. An alternative path would cover the situation in which a room becomes unavailable before the booking is completed.

The first prototype could contain a search page with filters for room capacity, location and equipment. A separate screen would display the selected room and booking details. The confirmation button would be placed where it is easy to find, and the application would display a clear message after the user completes the action.

The team could test this version with five students. Each participant would receive a task, such as finding a room for a group meeting at a specified time. The team would observe whether participants could find a suitable option, understand the availability information and complete the booking.

Five participants would provide initial feedback, but their results should not be treated as proof that every usability problem has been found. Nielsen and Landauer [16] proposed a mathematical model for estimating the proportion of usability problems detected by a number of independent evaluators. A simplified form of the model is given in Eq. (3):

$$P(i) = 1 - (1 - 0.31)^i \tag{3}$$

where:

- $P(i)$ is the estimated proportion of usability problems detected;
- i is the number of evaluators;
- 0.31 is the average probability of one evaluator detecting a particular problem in the model.

For five evaluators, the estimate is given in Eq. (4):

$$P(5) = 1 - (1 - 0.31)^5 \approx 0.844 \quad (4)$$

This gives an estimated detection rate of approximately 84% under the assumptions of the model. It should not be interpreted as a guarantee that five participants will identify 84% of all problems in any application. The actual result depends on the product, the tasks and the people taking part in the evaluation.

Suppose the team discovers that some participants overlook the search filters, while others are unsure whether their reservation has been confirmed. These findings would lead to specific changes. The filters could be made more noticeable, and the confirmation screen could display a clearer message with the booking details.

The team could then test the updated version and compare the results with the first test. Task completion, time spent and the number of difficulties observed would help show whether the changes improved the experience.

This example is hypothetical and does not present measured results. It demonstrates how a development team could use research findings to make design decisions and then check those decisions through testing. If the stages are disconnected, information collected at the beginning may never influence the final interface. When the results are used throughout development, the team has more opportunities to notice and correct problems.

VII. CONCLUSION

A software application can work correctly and still be difficult to use. The room-booking example illustrates how this can happen: students may have access to all the necessary functions but still struggle to find a room, understand its availability or complete a reservation.

The research question asked how user research, personas, user flows and prototype testing can help reduce these problems. The discussion suggests that their main value lies in how they are used together. Interviews and observation provide information about users' actual experiences. Personas help the team keep different user goals in mind, while user flows show what people need to do to complete a task. Prototypes then allow these decisions to be examined before the application is fully developed.

The example also shows why testing should not be treated as a final check performed only when the design is finished. A test may reveal that a filter is difficult to find or that users do not understand the confirmation message. Such findings can lead to changes in the interface and, if necessary, to a reconsideration of the original user flow.

Usability principles, including Fitts's law, Hick's law and Nielsen's heuristics, provide further guidance when making these changes. However, they cannot replace information about the people who will actually use the product.

Therefore, combining research, user flows, personas and prototype testing gives development teams a more informed way to make interface decisions. It does not guarantee that all usability problems will be avoided, but it makes it possible to identify many of them before they become part of a finished product. UX should be considered throughout software development, not added only after the technical work has been completed.

REFERENCES

[1] International Organization for Standardization. (2019). *ISO 9241-210:2019 Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. ISO. https://www.iso.org/standard/77520.html

[2] Hassenzahl, M., & Tractinsky, N. (2006). User experience – A research agenda. *Behaviour & Information Technology, 25*(2), 91–97. https://doi.org/10.1080/01449290500330331

[3] Beyer, H., & Holtzblatt, K. (1998). *Contextual design: Defining customer-centered systems*. Morgan Kaufmann.

[4] Nielsen, J. (2000). Why you only need to test with 5 users. *Nielsen Norman Group*. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[5] Cooper, A. (1999). *The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity*. Sams.

[6] Pruitt, J., & Grudin, J. (2003). Personas: Practice and theory. In *Proceedings of the 2003 Conference on Designing for User Experiences* (pp. 1–15). Association for Computing Machinery. https://dl.acm.org/doi/10.1145/997078.997089

[7] Garrett, J. J. (2011). *The elements of user experience: User-centered design for the web and beyond* (2nd ed.). New Riders.

[8] Houde, S., & Hill, C. (1997). What do prototypes prototype? In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), *Handbook of human-computer interaction* (2nd ed., pp. 367–381). Elsevier. https://doi.org/10.1016/B978-044481862-1/50082-0

[9] Rettig, M. (1994). Prototyping for tiny fingers. *Communications of the ACM, 37*(4), 21–27. https://doi.org/10.1145/175276.175288

[10] International Organization for Standardization. (2018). *ISO 9241-11:2018 Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts*. ISO. https://www.iso.org/standard/63500.html

[11] Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 152–158). Association for Computing Machinery. https://doi.org/10.1145/191666.191729

[12] Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. Basic Books.

[13] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*(6), 381–391. https://www.iso.org/standard/63500.html

[14] MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction, 7*(1), 91–139. https://doi.org/10.1207/s15327051hci0701_3

[15] Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology, 4*(1), 11–26. https://doi.org/10.1080/17470215208416600

[16] Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 206–213). Association for Computing Machinery. https://dl.acm.org/doi/10.1145/169059.169166